

⑯ BUNDESREPUBLIK
DEUTSCHLAND



DEUTSCHES
PATENTAMT

⑯ Offenlegungsschrift
⑯ DE 44 16 881 A 1

⑯ Int. Cl. 5:
G 06 F 15/80

⑯ Innere Priorität: ⑯ ⑯ ⑯

13.05.93 DE 43 16 036.0

⑯ Anmelder:

Vorbach, Martin, 78137 Karlsruhe, DE

⑯ Vertreter:

Zahn, R., Dipl.-Ing., Pat.-Anw., 76229 Karlsruhe

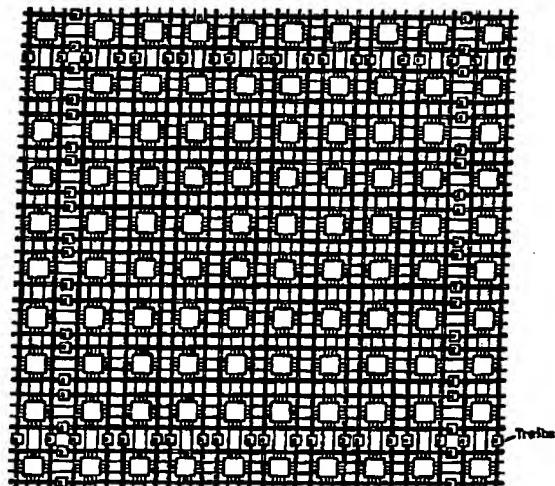
⑯ Aktenzeichen: P 44 16 881.0
⑯ Anmeldetag: 13. 5. 94
⑯ Offenlegungstag: 17. 11. 94

⑯ Erfinder:
gleich Anmelder

Prüfungsantrag gem. § 44 PatG ist gestellt

⑯ Datenverarbeitungseinrichtung

- ⑯ In Verbindung mit einer Datenverarbeitungseinrichtung, wobei ein (im folgenden Datenflußprozessor - DFP - genannter) integrierter Schaltungskreis (Chip) mit einer Vielzahl insbesondere orthogonal zueinander angeordneter homogener strukturierter Zellen mit je einer Mehrzahl jeweils logisch gleicher und strukturell identisch angeordneter Bausteine vorgesehen ist, dessen Zellen zeilen- und spaltenweise, gegebenenfalls gruppenweise zusammengefaßt, mit Ein-/Ausgangsanschlüssen des integrierten Schaltungskreises verbunden sind, ist erfindungsgemäß den Zellen eine Ladelogik zugeordnet, über die sie je für sich und gegebenenfalls gruppenweise zusammengefaßt so programmierbar (konfigurierbar) sind, daß beliebige logische Funktionen und/oder Vernetzungen untereinander verifizierbar sind, und zwar derart, daß eine Manipulation der DFP-Konfiguration während des Betriebes (oder zur Laufzeit), d. h. die Modifikation funktioneller Teile (MACROS) des DFPs erfolgen kann, ohne daß andere funktionelle Teile gehalten werden müssen oder in ihrer Funktion beeinträchtigt werden.



DE 44 16 881 A 1

Die folgenden Angaben sind den vom Anmelder eingesetzten Unterlagen entnommen

BUNDESDRUCKEREI 09.94 408 046/623

DE 44 16 881 A 1

DE 44 16 881 A1

1

Beschreibung

Die vorliegende Erfindung bezieht sich auf eine Datenverarbeitungseinrichtung, d. h. eine Hardwareeinheit zur logischen Manipulation (Verknüpfung) von in binärer Form vorliegenden Daten (Informationen).

Derartige Datenverarbeitungseinrichtungen sind mittlerweile lange bekannt und sie haben bereits breite Anwendung und Anerkennung gefunden. Die prinzipielle Aufbau- und Arbeitsstruktur der bekannten Datenverarbeitungseinrichtungen ist in etwa so zu definieren, daß eine arithmetisch-logische Verknüpfungseinheit vorgesehen ist, in der die zu verknüpfenden Daten einer programmtechnischen Anweisung (Software) zufolge verarbeitet werden. Die Daten werden dabei über ein Steuerwerk in mehr oder weniger komplexen Adressierungsvorgängen entsprechend abgerufen und zunächst in Arbeitsregistern bereitgestellt; nach der logischen Verknüpfung werden dann die neuen Daten in einer vorgegebenen Speicherstelle wieder abgelegt. Die arithmetisch-logische Verknüpfungseinheit besteht dabei aus logischen Verknüpfungsbausteinen (Gatter, Glieder), die jeweils so miteinander gekoppelt sind, daß die zu manipulierenden Daten der zugrunde liegenden Software entsprechend den vier Grundrechenarten gemäß logisch verarbeitet werden.

Es ist leicht nachzuvollziehen, daß auf der Basis der bekannten Strukturen relativ viel Rechenzeit dafür erforderlich ist, die zu manipulierenden Daten auszulesen, und die Arbeitsregister zu überführen, den spezifischen Logikbausteinen in der arithmetisch-logischen Verknüpfungseinheit zuzuleiten und schließlich wieder abzuspeichern. Es ist ferner einsichtig, daß die Hardwarestruktur der arithmetisch-logischen Verknüpfungseinheit insoweit nicht als optimal betrachtet werden kann, als schließlich die hardwaremäßig vorhandenen integrierten logischen Bausteine stets nur in ein und derselben Art und Weise im Gesamtsystem aktiv benutzt werden. Ebenso wird durch strikte Hardwarevorgabe ein Aneinanderreihen von Funktionen in sogenannten Pipelines sehr erschwert oder eingeschränkt, was zwangsläufig ein häufiges Registerumladen zwischen Arbeitsregistern und Rechenwerk bedeutet. Derartige Bausteine sind des weiteren nur schlecht kaskadierbar und erfordern dann sehr viel Programmierarbeit.

Ein zusätzlicher Vorteil der vorliegenden Erfindung liegt darin, daß eine über einen weiten Raum skalierbare Parallelität zur Verfügung steht. Hierbei wird eine Basis zum schnellen und flexiblen Aufbau von neuronalen Strukturen geschaffen, wie die bis dato lediglich mit erheblichem Aufwand simuliert werden können.

Die der vorliegenden Erfindung zugrunde liegende Aufgabe besteht darin, eine im folgenden Datenflußprozessor (DFP) genannte Datenverarbeitungseinrichtung anzugeben, bei der eine höhere beziehungsweise bessere Flexibilität der Gesamtstruktur und des Datenflusses sowie der Pipelining- und Kaskadiermöglichkeiten zu einer Erhöhung der Rechnerbeziehungsweise Verknüpfungsleistung führt.

Außer dem Einsatz als reiner Datenflußprozessor, soll der DFP folgende weitere Aufgaben erfüllen können:

- Einsatz als universeller Baustein zum Aufbau von herkömmlichen Rechnern, wobei der Aufbau einfacher und billiger werden soll.
- Einsatz in neuronalen Netzen.

2

Diese Aufgabe wird dadurch gelöst, daß ein integrierter Schaltkreis (Chip) mit einer Vielzahl insbesondere orthogonal zueinander angeordneter Zellen mit je einer Mehrzahl jeweils logisch gleicher und strukturell identisch angeordneter Zellen vorgesehen ist deren Anordnung, sowie die interne Busstruktur, zur Erleichterung der Programmierung äußerst homogen ist. Dennoch ist es denkbar innerhalb eines Datenflußprozessors Zellen mit verschiedenen Zelllogiken und Zellstrukturen unterzubringen, um so die Leistungsfähigkeit zu erhöhen, indem zum Beispiel für Speicheransteuerungen andere Zellen als für arithmetische Operationen existieren. Insbesondere kann für neuronale Netze eine gewisse Spezialisierung von Vorteil sein. Den Zellen ist eine Ladelogik zugeordnet, über die die Zellen je für sich und gegebenenfalls gruppenweise in sogenannte MACROS zusammengefaßt so programmierbar sind, daß einerseits beliebige logische Funktionen, andererseits aber auch die Verknüpfung der Zellen untereinander in weiten Bereichen verifizierbar sind. Dies wird erreicht indem jeder einzelnen Zelle ein gewisser Speicherplatz zur Verfügung steht, in dem die Konfigurationsdaten abgelegt sind. Anhand dieser Daten werden Multiplexer oder Transistoren in der Zelle beschaltet um die jeweilige Zellfunktion zu gewährleisten (siehe Fig. 12).

Mit anderen als im Patentanspruch 1 gebrauchten Worten besteht der Kern der vorliegenden Erfindung darin, einen Datenflußprozessor vorzuschlagen, der zellular aufgebaut ist und dessen Zellen über eine externe Ladelogik im arithmetisch-logischen Sinne quasi beliebig neu konfiguriert werden können. Dabei ist es von äußerster Notwendigkeit, daß die betreffenden Zeilen einzeln und ohne Beeinflussung der übrigen Zeilen oder gar einer Stilllegung des gesamten Bausteins umkonfiguriert werden können. Der Datenflußprozessor gemäß der vorliegenden Erfindung kann so während eines ersten Arbeitszyklus als Addierer und während eines späteren Arbeitszyklus als Multiplizierer "programmiert" werden, wobei die Anzahl der für die Addition beziehungsweise die Multiplikation erforderlichen Zellen durchaus unterschiedlich sein können. Dabei bleibt die Plazierung der bereits geladenen MACROS erhalten; der Ladelogik beziehungsweise dem Compiler obliegt es, das neu zu ladende MACRO innerhalb der freien Zellen zu partitionieren (d. h. das zu ladende MACRO so zu zerlegen, daß es sich optimal einfügen läßt). Die Ablaufsteuerung des Programms wird dabei von der Ladelogik übernommen, indem sie gemäß dem momentan ausgeführten Programmabschnitt die entsprechenden MACROS in den Baustein lädt, wobei der Ladevorgang von der später beschriebenen Synchronisationslogik mitgesteuert wird, indem sie den Zeitpunkt des Umladens festlegt. Daher entspricht der DFP nicht der bekannten von-Neumann-Architektur, da die Daten- und Programmspeicher getrennt sind. Dies bedeutet jedoch gleichzeitig eine höhere Sicherheit, da fehlerhafte Programme keinen CODE, sondern lediglich DATEN zerstören können.

Um den Datenflußprozessor eine arbeitsfähige Struktur zu geben, werden einige Zellen, und zwar unter anderem die Eingabe-/Ausgabefunktionen (I/O) und Speichermanagementfunktionen vor dem Laden der Programme geladen und bleiben für gewöhnlich während der gesamten Laufzeit konstant. Dies ist erforderlich um den Datenflußprozessor an seine Hardwareumgebung anzupassen. Die übrigen Zellen werden zu sogenannten MACROS zusammengefaßt und können während der Laufzeit nahezu beliebig und ohne Beeinflussung der

DE 44 16 881 A1

3

Nachbarzellen umkonfiguriert werden. Dazu sind die Zellen einzeln und direkt adressierbar.

Um die Umstrukturierung (das Umladen) der Zellen oder MACROs mit der Ladelogik zu synchronisieren, kann – wo notwendig, da nur Umgeladen werden darf, wenn die MACROs mit ihrer alten Tätigkeit fertig sind – eine Synchronisationsschaltung als MACRO auf dem Datenflußprozessor untergebracht werden, die die entsprechenden Signale an die Ladelogik absendet. Hierzu kann eventuell eine Modifikation der gewöhnlichen MACROs von Nöten sein, da diese dann der Synchronisations-Schaltung Zustandsinformationen zur Verfügung stellen müssen.

Diese Zustandsinformationen signalisieren der Synchronisationslogik für gewöhnlich, daß einzelne MACROs ihre Aufgabe erledigt haben, was aus programmiertechnischer Sicht zum Beispiel die Terminierung einer Prozedur oder das Erreichen der Terminierungsbedingung einer Schleife bedeuten kann. D.h. das Programm wird an einer anderen Stelle fortgesetzt und die die Zustandsinformation absendenden MACROs können umgeladen werden. Zudem kann es von Interesse sein, daß die MACROs in einer bestimmten Reihenfolge umgeladen werden. Hierzu kann eine Wertung der einzelnen Synchronisations-Signale durch einen Prioritätsdecoder erfolgen. Eine derartige – einfache – Logik ist in Fig. 13 gezeichnet. Die Logik besitzt sieben Eingangssignale durch die die sieben MACROs ihre Zustandsinformation abgeben. In diesem Fall soll 0 für "in Arbeit" und 1 für "fertig" stehen. Die Logik besitzt drei Ausgangssignale, die an die Ladelogik geführt werden, wobei der Zustand 000 als Ruhezustand gilt. Liegt ein Signal an einem der sieben Eingänge an, so findet eine Dezimal-Binär-Umsetzung statt, so wird zum Beispiel Sync6 als 110 dargestellt, was der Ladelogik anzeigt, daß das MACRO, welches Sync6 bedient, seine Aufgabe beendet hat. Liegen gleichzeitig mehrere Synchronisations-Signale am Eingang an, so gibt die Synchronisationsschaltung das Signal mit der höchsten Priorität an die Ladelogik weiter; liegen zum Beispiel Sync0, Sync4 und Sync6 an, so reicht die Syncronisations-Schaltung zunächst Sync6 an die Ladelogik weiter. Nachdem die entsprechenden MACROs umgeladen sind und somit Sync6 nicht mehr anliegt wird Sync4 weitergeleitet usw. Zur Verdeutlichung dieses Prinzips kann der Standard-TTL-Baustein 74148 in Betracht gezogen werden.

Über die Ladelogik kann der Datenflußprozessor jeweils optimal und gegebenenfalls dynamisch auf eine zu lösende Aufgabe eingestellt werden. Damit ist zum Beispiel der große Vorteil verbunden, daß neue Normen oder dergleichen einzig und allein durch eine Umprogrammierung des Datenflußprozessors umgesetzt werden können und nicht – wie bisher – einen Austausch mit entsprechendem Anfall von Elektronikschratt bedingen.

Die Datenflußprozessoren sind untereinander kaskadierbar, was zu einer beinahe beliebigen Erhöhung des Parallelisierungsgrades, der Rechenleistung, sowie der Netzgröße in neuronalen Netzen führt. Besonders wichtig ist hier eine klare homogene Verbindung der Zellen mit den Ein-/Ausgangs-Pins (IO-Pin) der Datenflußprozessoren, um möglichst keine Einschränkungen auf die Programme zu haben.

In Fig. 14 ist zum Beispiel die Kaskadierung von vier DFPs gezeigt. Sie erscheinen der Umgebung wie ein großer homogener Baustein (Fig. 15). Prinzipiell sind damit zwei Kaskadierungsmethoden denkbar:

4

a) Nur die lokalen Verbindungen zwischen den Zellen werden herausgeführt, was im vorliegenden Beispiel zwei IO-Pins pro Kantenzelle und vier IO-Pins pro Eckzelle bedeutet. Allerdings hat der Compiler/Programmierer zu beachten, daß die globalen Verbindungen nicht herausgeführt werden, wodurch die Kaskadierung nicht vollständig homogen ist (Globale Verbindungen zwischen mehreren Zellen, für gewöhnlich zwischen einer kompletten Zellenreihe oder -spalte – siehe Fig. 6 –; lokale Verbindungen existieren nur zwischen zwei Zellen). Fig. 16a zeigt den Aufbau innerhalb eines DFPs, Fig. 17a zeigt die daraus resultierende Kaskadierung von mehreren DFPs (drei gezeichnet).

b) Die lokalen und globalen Verbindungen werden herausgeführt, was die Anzahl der benötigten Treiber/IO-Pins und Leitungen drastisch erhöht, in unserem Beispiel auf sechs IO-Pins pro Kantenzelle und zwölf IO-Pins pro Eckzelle. Dadurch ist eine vollständige Homogenität bei der Kaskadierung gegeben.

Da die globalen Verbindungen insbesondere bei Verwendung der Kaskadierungstechnik b) sehr lang werden können, kann der unangenehme Effekt auftreten, daß die Zahl der globalen Verbindungen nicht ausreicht, da bekanntlich jede Verbindung nur von einem Signal genutzt werden kann. Um diesen Effekt zu minimieren, kann nach einer gewissen Länge der globalen Verbindungen ein Treiber eingeschleift werden. Der Treiber hat zum einen eine Verstärkung des Signals zur Aufgabe, die bei langen Strecken und entsprechend hohen Lasten, unbedingt erforderlich ist; zum anderen kann der Treiber in Tristate gehen und damit das Signal unterbrechen. Dadurch können die Abschnitte links und rechts, beziehungsweise oberhalb und unterhalb des Treibers von verschiedenen Signalen genutzt werden, sofern der Treiber in Tristate ist, ansonsten wird ein Signal durchgeschleift. Wichtig ist hierbei, daß die Treiber der einzelnen globalen Leitungen auch einzeln angesteuert werden können, d. h. ein globales Signal kann unterbrochen sein, das Nachbarsignal ist jedoch durchgeschleift. Somit können auf einer globalen Verbindung durchaus abschnittsweise verschiedene Signale anliegen, während die globale Nachbarverbindung tatsächlich global von ein und demselben Signal verwendet wird (vergleiche Fig. 22).

Zur besseren Kommunikation zwischen den Datenflußprozessoren und der Ladelogik können sogenannte Shared-Memories eingesetzt werden. So können zum Beispiel Programme von einer Festplatte, die im IO-Bereich eines Datenflußprozessors liegt zur Ladelogik durchgereicht werden, indem die Datenflußprozessoren die Daten von der Platte in den Shared-Memory schreiben und die Ladelogik sie dort abholt. Dies ist besonders wichtig, da hier, wie bereits erwähnt, keine von-Neumann- sondern eine Harvardarchitektur vorliegt. Ebenso sind die Shared-Memories von Vorteil, wenn Konstanten, die im Programm – das im Speicherbereich der Ladelogik liegt – definiert sind, mit Daten – die im Speicherbetrieb der Datenflußprozessoren liegen – verknüpft werden sollen.

Weiterbildungen der vorstehend definierten und umschriebenen Erfahrung sind Gegenstand der Unteransprüche.

Eine besondere Verwendung des erfundungsgemäßen Datenflußprozessors ist darin zu sehen, daß er in Verbindung mit geeigneten Ein-/Ausgabe-Einheiten einer-

DE 44 16 881 A1

5

6

seits und einem Speicher andererseits die Basis für einen kompletten (komplexen) Rechner bilden kann. Dabei kann ein Großteil der IO-Funktionen als MACROS auf dem Datenflußprozessor implementiert werden und es brauchen momentan lediglich Spezialbausteine (Ethernet-Treiber, VRAMS...) extern zugefügt zu werden. Bei einer Normänderung oder Verbesserung muß dann wie bereits angedeutet nur das MACRO softwareseitig gewechselt werden; ein Eingriff in die Hardware ist nicht notwendig. Es bietet sich hier an, einen IO-(Eingabe-/Ausgabe-) Stecker festzulegen, über welchen dann die Zusatzausteine angeschlossen werden können.

Fig. 20 zeigt den stark vereinfachten Aufbau eines heute üblichen Rechners. Durch den Einsatz eines DFP-Bausteins können erhebliche Teile eingespart werden (Fig. 21), wobei die entsprechenden herkömmlichen Baugruppen (CPU, Speicherverwaltung, SCSI-, Tastatur- und Videointerface, sowie der parallelen und seriellen Schnittstellen) als MACROS in die kaskadierten DFPs abgelegt werden. Nur die durch einen DFP nicht nachbildungsbaren Teile wie Speicher und Leitungstreiber mit nicht TTL-Pegeln oder für hohe Lasten müssen extern zugeschaltet werden. Durch die Verwendung des DFPs ist eine günstige Produktion gegeben, da ein und derselbe Baustein sehr häufig verwendet wird, das Layout der Platine ist durch die homogene Vernetzung entsprechend einfach. Zudem wird der Aufbau des Rechners durch die Ladelogik bestimmt, die hier für gewöhnlich nur zu Beginn der Abarbeitung (nach einem Reset) das DFP-Array lädt, wodurch eine günstige Fehlerkorrektur- und Erweiterungsmöglichkeit gegeben ist. Ein derartiger Rechner kann insbesondere mehrere verschiedene Rechnerstrukturen simulieren, indem einfach der Aufbau des zu simulierenden Rechners in das DFP-Array geladen wird. Zu bemerken ist, daß hierbei der DFP nicht in seiner Funktion als DFP arbeitet sondern lediglich ein hochkomplexes und frei programmierbares Logikarray zur Verfügung stellt, sich hierbei jedoch von herkömmlichen Bausteinen in seiner besonderen guten Kaskadierbarkeit unterscheidet.

Ein weiteres Einsatzgebiet des Bausteins ist der Aufbau großer neuronaler Netze. Sein besonderer Vorzug liegt hierbei in seiner hohen Gatterdichte, seiner ausgezeichneten Kaskadierbarkeit, sowie seiner Homogenität. Ein Lernvorgang, der eine Änderung einzelner axiomatischer Verbindungen beinhaltet ist auf üblichen Bausteinen ebenso schlecht durchführbar, wie der Aufbau großer homogener und gleichzeitig flexibler Zellstrukturen. Die dynamische Umkonfigurierbarkeit ermöglicht erstmalig die optimale Simulation von Lernvorgängen.

Die vorliegende Erfindung wird im folgenden anhand der weiteren Figuren näher erläutert. Insgesamt zeigen

Fig. 1 ein Schaltsymbol für einen 8-Bit-Addierer;

Fig. 2 ein Schaltsymbol für einen aus acht 1-Bit-Addierern bestehenden 8-Bit-Addierer nach Fig. 1;

Fig. 3 eine logische Struktur eines 1-Bit-Addierers entsprechend Fig. 2;

Fig. 4 eine Zellenstruktur des 1-Bit-Addierers entsprechend Fig. 3;

Fig. 5 einen der Zellenstruktur nach Fig. 1 entsprechend aufgebauten 8-Bit-Addierer;

Fig. 6 ein aus vier Zellen bestehendes unprogrammiertes SUBMACRO X (analog einem 1-Bit-Addierer gemäß Fig. 4 beziehungsweise Fig. 5) mit den erforderlichen Leitungsanschlüssen;

Fig. 7 einen Teilausschnitt eines integrierten Schaltkreises (Chip) mit einer Vielzahl von Zellen und einem

separierten SUBMACRO X gemäß Fig. 6;

Fig. 8 einer integrierten Schaltkreis (Chip) mit einer Orthogonalstruktur einer quasi beliebigen Vielzahl von Zellen und einer extern zugeordneten Ladelogik;

5 Fig. 9 ein erstes Ausführungsbeispiel einer Mehrzahl miteinander zu einem Rechenwerk gekoppelter integrierter Schaltkreise (Datenflußprozessor) nach Fig. 8;

Fig. 10 ein zweites Ausführungsbeispiel einer Mehrzahl miteinander zu einem Rechenwerk gekoppelter integrierter Schaltkreise (Datenflußprozessor) nach Fig. 8;

Fig. 11 ein Ausführungsbeispiel eines MACRO zur Addition zweier Zahlenreihen;

Fig. 12 einen beispielhaften Aufbau einer Zelle mit 15 Multiplexern zur Auswahl der jeweiligen logischen Bausteine;

Fig. 13 eine zum Beispiel mit einem Standard-TTL-Baustein 74148 ausgeführte Synchronisationslogik;

Fig. 14 die Kaskadierung von vier DFPs, wobei die 20 Verbindung zwischen den IO-Pins nur schematisch dargestellt sind (tatsächlich bedeutet eine gezeichnete Verbindung eine Mehrzahl von Leitungen);

Fig. 15 die durch die Kaskadierung erreichte Homogenität;

25 Fig. 16a die Struktur der E/A-Zellen, wobei die globalen Verbindungen nicht herausgeführt werden,

Fig. 16b die Struktur der E/A-Zellen, jedoch mit herausgeführten globalen Verbindungen;

30 Fig. 17a die aus Fig. 16a resultierende Kaskadierung, wobei eine Eckzelle, sowie die zwei mit ihr kommunizierenden Treiberzellen der kaskadierten Bausteine (vergleiche hierzu Fig. 14) gezeichnet sind;

Fig. 17b die aus Fig. 16b resultierende Kaskadierung, wobei eine Eckzelle, sowie die zwei mit ihr kommunizierenden Treiberzellen der kaskadierten Bausteine (vergleiche hierzu Fig. 14) gezeichnet sind;

35 Fig. 18a eine Multiplikationsschaltung (vergleiche Fig. 11a);

Fig. 18b die interne Struktur des DFPs nach dem La-40 den (vergleiche Fig. 11b);

Fig. 19c die Arbeitsweise des DFPs im Speicher, so- wie die Zustände der Zähler 47, 49;

Fig. 19 eine Kaskadenschaltung, wobei der Addierer aus Fig. 11 und der Multiplizierer aus Fig. 18 zur Steige-45 rung der Rechenleistung hintereinander geschaltet sind;

Fig. 20 den stark schematisierten Aufbau eines herkömmlichen Rechners;

Fig. 21 den möglichen Aufbau desselben Rechners mit Hilfe eines Arrays aus kaskadierten DFPs;

50 Fig. 22 einen Ausschnitt mit eingezeichneten (Lei-tungs-)Treibern eines DFPs.

In Fig. 1 ist ein Schaltsymbol eines 8-Bit-Addierers dargestellt. Das Schaltsymbol besteht aus einem quadratischen Baustein 1 mit acht Eingängen A 0...7 für ein erstes Datenwort A und acht Eingängen B 0...7 für ein zweites (zu addierendes) Datenwort B. Die jeweils acht Eingänge A_i, B_i werden ergänzt durch einen weiteren Eingang C_{in} über den dem Baustein 1 gegebenenfalls ein Übertrag zugeleitet wird. Der Baustein 1 hat funktions- und bestimmungsgemäß acht Ausgänge S 0...7 für binären Summanden und einen weiteren Ausgang C_{out} für den gegebenenfalls bestehenden Übertrag.

65 Das in Fig. 1 dargestellte Schaltsymbol ist in Fig. 2 als Anordnung sogenannter SUBMACROS dargestellt. Diese SUBMACROS 2 bestehen je aus einem 1-Bit-Addierer 3 mit je einem Eingang für die entsprechenden Bits des Datenworts und einem weiteren Eingang für ein

DE 44 16 881 A1

7

8

Übertragsbit. Die 1-Bit-Addierer 3 weisen darüberhinaus einen Ausgang für den Summanden und einen Ausgang für den Übertrag Üaus auf.

In Fig. 3 ist die binäre Logik eines 1-Bit-Addierers beziehungsweise eines SUBMACROS 2 nach Fig. 2 dargestellt. Analog zu Fig. 2 weist diese Schaltlogik je einen Eingang A_i , B_i für die konjugierten Bits der zu verknüpfenden Daten auf; ferner ist ein Eingang \bar{U}_{ein} für den Übertrag vorgesehen. Diese Bits werden den dargestellten Verbindungen beziehungsweise Verknüpfungen entsprechend in zwei ODER-Gliedern 5 und drei NAND-Gliedern 6 verknüpft, so daß am Ausgangsanschluß S_i und am Ausgang für den Übertrag \bar{U}_{aus} die einem Volladdierer entsprechenden Verknüpfungsergebnisse (S_i , \bar{U}_{aus}) anstehen.

Die Erfindung setzt da ein, wo es — wie in Fig. 4 dargestellt — darum geht, das in Fig. 3 gezeigte SUBMACRO 2 oder eine oder mehrere beliebige Funktion(en) in geeigneter Weise in einer Zellstruktur zu implementieren. Dies geschieht auf der Grundlage logisch und strukturell identischer Zellen 10, deren einzelne logische Bausteine der auszuführenden Verknüpfungsfunktion entsprechend miteinander gekoppelt werden, und zwar mittels der noch zu beschreibenden Ladelogik. Gemäß der in Fig. 4 gezeigten, von der Schaltlogik nach Fig. 3 abgeleiteten Verknüpfungslogik für einen 1-Bit-Addierer sind je zwei Zellen 10.1, 10.2 bezüglich der logischen Bausteine insoweit gleich, daß jeweils ein ODER-Glied 5 und ein NAND-Glied 6 aktiviert sind. Eine dritte Zelle 10.3 wird nur als Leitungszelle (Leiterbahnzelle) benutzt und die vierte Zelle 10.4 ist bezüglich des dritten NAND-Gliedes 6 aktiv geschaltet. Das aus den vier Zellen 10.1 ... 10.4 bestehende SUBMACRO 2 steht somit stellvertretend für einen 1-Bit-Addierer, d. h. ein 1-Bit-Addierer einer Datenverarbeitungseinrichtung gemäß der vorliegenden Erfindung wird über vier entsprechend programmierte (konfigurierte) Zellen 10.1 ... 10.4 verifiziert. (Der Vollständigkeit halber soll angemerkt werden, daß die einzelnen Zellen ein erheblich umfangreicheres Netzwerk von logischen Bausteinen, sprich Verknüpfungsgliedern, und Invertoren aufweist, die jeweils dem aktuellen Befehl der Ladelogik zufolge aktiv geschaltet werden können. Neben den logischen Bausteinen ist auch ein dichtes Netz von Verbindungsleitungen zwischen den jeweils benachbarten Bausteinen und zum Aufbau von zeilen- und spaltenweisen Busstrukturen zur Datenübertragung andererseits vorgesehen, so daß über eine entsprechende Programmierung seitens der Ladelogik quasi beliebige logische Verknüpfungsstrukturen implementiert werden können).

Der Vollständigkeit halber ist in Fig. 5 der Zellaufbau eines 8-Bit-Addierers in seiner Gesamtheit dargestellt. Die in Fig. 5 gezeigte Struktur entspricht insoweit der nach Fig. 2, wobei die in Fig. 2 symbolisch als SUBMACROS 2 dargestellten 1-Bit-Addierer jeweils durch eine vier-zellige Einheit 10.1 ... 10.4 ersetzt sind. Bezug auf den erfundungsgemäßen Datenflußprozessor bedeutet dies, daß zweiunddreißig Zellen der zur Verfügung stehenden Gesamtheit von Zellen einer zellular mit logisch identischem Layout gefertigten Schaltungsplatine seitens der Ladelogik so angesteuert und konfiguriert beziehungsweise programmiert werden, daß diese zweiunddreißig Zellen ein 8-Bit-Addierer bilden.

In der Darstellung nach Fig. 5 ist über eine strichpunktierte Umrahmung ein SUBMACRO "X" zeichnerisch separiert, das letztlich als aus vier einem 1-Bit-Addierer entsprechend programmierten Zellen (10 gemäß

Fig. 4) bestehende Untereinheit zu betrachten ist.

Das in Fig. 5 separierte SUBMACRO "X" ist in Fig. 6 als Teil eines integrierten Schaltkreises (Chip) 20 gemeinsam mit Leitungs- und Datenanschlüssen dargestellt. Das SUBMACRO "X" besteht aus den vier Zellen 10 die entsprechend der orthogonalen Struktur je Seite vier Datenanschlüsse (also insgesamt sechzehn Datenanschlüsse je Zelle) aufweisen. Die Datenanschlüsse verbinden jeweils benachbarte Zellen, so daß ersichtlich wird, wie beispielsweise eine Dateneinheit von Zelle zu Zelle durchgeschleust wird. Die Ansteuerung der Zellen 10 erfolgt einerseits über sogenannte lokale Steuerungen, das sind lokale Leitungen, die mit allen Zellen verbunden sind, und andererseits über sogenannte globale Leitungen, d. h. Leitungen, die über den gesamten integrierten Schaltkreis (Chip) 20 geführt sind.

In Fig. 7 ist ein vergrößerter Ausschnitt eines integrierten Schaltkreises 20 dargestellt, der mit einem orthogonalen Raster von Zellen 10 belegt ist. Wie in Fig. 7 angedeutet kann so zum Beispiel eine Gruppe von vier Zellen 10 als SUBMACRO "X" ausgewählt und dem 1-Bit-Addierer entsprechend Fig. 4 gemäß programmiert beziehungsweise konfiguriert werden.

Ein vollständiger integrierter Schaltkreis (Chip) 20 ist in Fig. 8 dargestellt. Dieser integrierte Schaltkreis 20 besteht aus einer Vielzahl im orthogonalen Raster angeordneter Zellen 10 und weist an seinen Außenkanten eine entsprechende Anzahl von Leitungsanschlüssen (Pins) auf, über die Signale, insbesondere Ansteuersignale und Daten zugeführt und weitergeleitet werden können. In Fig. 8 ist wiederum das SUBMACRO "X" gemäß Fig. 5/Fig. 6 abgegrenzt; darüberhinaus sind auch weitere SUBMACROS separiert, die spezifischen Funktionen und Vernetzungen entsprechend zu Untereinheiten zusammengefaßt sind. Dem integrierten Schaltkreis (Chip) 20 ist eine Ladelogik 30 zugeordnet beziehungsweise übergeordnet, über die der integrierte Schaltkreis 20 programmiert und konfiguriert wird. Die Ladelogik 30 teilt letztlich dem integrierten Schaltkreis 20 mit, wie er arithmetisch-logisch zu arbeiten hat. Bezugnehmend auf die Fig. 1 bis 5 ist in Fig. 8 einerseits das SUBMACRO "X" entsprechend Fig. 4 und Fig. 5 hervorgehoben; andererseits ist auch ein MACRO "Y" entsprechend Fig. 1 und Fig. 2 angezeichnet, das als Einheit einem 8-Bit-Addierer entspricht.

Anhand von Fig. 9 beziehungsweise Fig. 10 soll im folgenden eine Rechnerstruktur beschrieben werden, die auf den im vorstehenden definierten und erläuterten integrierten Schaltkreis 20 aufbaut.

Gemäß dem in Fig. 9 dargestellten ersten Ausführungsbeispiel ist — analog zur Anordnung der Zellen — im Orthogonalraster eine Mehrzahl von integrierten Schaltkreisen 20 angeordnet, deren jeweils benachbarte über lokale BUS-Leitungen 21 miteinander gekoppelt beziehungsweise vernetzt sind. Die — beispielsweise aus sechzehn integrierten Schaltkreisen 20 bestehende — Rechnerstruktur weist Ein-/Ausgangsleitungen IO auf, über die der Rechner quasi mit der Außenwelt in Verbindung steht, d. h. korrespondiert. Der Rechner gemäß Fig. 9 weist ferner einen Speicher 22 auf, der dem dargestellten Ausführungsbeispiel entsprechend aus zwei separierten Speichern, zusammengesetzt aus jeweils RAM, ROM sowie einem Dual-Ported RAM als shared memory zu der Ladelogik geschaltet, besteht, die gleichermaßen als Schreib-Lese-Speicher oder auch nur als Lese-Speicher realisiert sein können. Der soweit beschriebenen Rechnerstruktur ist die Ladelogik 30 beziehungsweise übergeordnet, mittels der die inte-

DE 44 16 881 A1

9

10

grierten Schaltkreise (Datenflußprozessor) 20 programmiert und konfiguriert und vernetzt werden.

Die Ladelogik 30 baut auf einem Transputer 31, d. h. einem Prozessor mit mikrocodiertem Befehlssatz auf, dem seinerseits ein Speicher 32 zugeordnet ist. Die Verbindung zwischen dem Transputer 31 und dem Datenflußprozessor basiert auf einer Schnittstelle 33 für die sogenannten Ladedaten, d. h. die Daten die den Datenflußprozessor aufgaben spezifisch programmieren und konfigurieren und einer Schnittstelle 34 für den bereits genannten Rechnerspeicher 22, d. h. den Shared-Memory-Speicher.

Die in Fig. 9 dargestellte Struktur stellt so einen kompletten Rechner dar, der über die Ladelogik 30 jeweils fall- beziehungsweise aufgaben spezifisch programmiert und konfiguriert werden kann. Der Vollständigkeit halber sei noch angemerkt, daß – wie in Verbindung mit der Ladelogik 30 über Pfeile angedeutet – mehrere dieser Rechner vernetzt, d. h. miteinander gekoppelt werden können.

Ein weiteres Ausführungsbeispiel einer Rechnerstruktur ist in Fig. 10 dargestellt. Im Unterschied zu Fig. 9 sind dabei neben den lokalen BUS-Leitungen zwischen den benachbarten integrierten Schaltkreisen 20 noch übergeordnete zentrale BUS-Leitungen 23 vorgesehen, um zum Beispiel spezifische Ein- beziehungsweise Ausgangsprobleme lösen zu können. Auch der Speicher 22 (Shared-Memory) ist über zentrale BUS-Leitungen 23 mit den integrierten Schaltkreisen 20 verbunden, und zwar wie dargestellt jeweils mit Gruppen dieser integrierten Schaltkreise. Die in Fig. 10 dargestellte Rechnerstruktur weist die gleiche Ladelogik 30 auf, wie sie anhand von Fig. 9 erläutert wurde.

In Verbindung mit Fig. 11a soll eine auf erfundungsge- mäßen Datenflußprozessoren aufgebaute Additions- schaltung erläutert werden. Ausgegangen wird von zwei Zahlenreihen An und Bn für sämtliche n zwischen 0 und 9; die Aufgabe besteht darin, die Summe Ci = Ai + Bi zu bilden, wobei der Index i die Werte 0 <= n < 9 annehmen kann.

Bezugnehmend auf die Darstellung nach Fig. 11a ist die Zahlenreihe An in einem ersten Speicher RAM1 abgespeichert und zwar zum Beispiel ab einer Speicher- adresse 1000h; die Zahlenreihe Bn ist in einem Speicher RAM2 an einer Speicheradresse 0dfa0h abgespeichert; die Summe Cn wird in den RAM1 eingeschrieben und zwar unter der Adresse 100ah.

Es ist ein weiterer Zähler 49 zugeschaltet, der lediglich die einzelnen durch die Steuerschaltung freigegebenen Taktzyklen hochzählt. Dies soll im Weiteren zur Verdeutlichung der Umkonfigurierbarkeit einzelner MACROs ohne Beeinflussung der an der Umkonfigurierung nicht beteiligten MACROs dienen.

Fig. 11a zeigt zunächst die eigentliche Additions- schaltung 40, die aus einem ersten Register 41 zur Aufnahme der Zahlenreihe An und einem zweiten Register 42 zur Aufnahme der Zahlenreihe Bn besteht. Den beiden Registern 41/42 ist ein 8-Bit-Addierer entsprechend dem in Fig. 1 dargestellten MACRO 1 nachgeschaltet. Der Ausgang des MACRO 1 führt über eine Treiberschaltung 43 zurück zum Speicher RAM1. Die Takt- beziehungsweise Zeitsteuerung der Additionsschaltung 40 erfolgt über eine von einem Taktgenerator T ange- steuerte Zeitsteuerung (STATEMACHINE) 45, die mit den Registern 41, 42 und der Treiberschaltung 43 ver- bunden ist.

Die Additionsschaltung 40 wird funktional durch eine Adressschaltung 46 zur Generierung der Adressdaten für

die abzuspeichernden Additionsergebnisse ergänzt. Die Adressschaltung 46 besteht ihrerseits aus drei MACROs 1 (gemäß Fig. 1) zur Bildung der Adressdaten, wobei diese MACROs 1 wie folgt geschaltet sind: Über jeweils 5 einen Eingang werden die zu verknüpfenden Adressen für An, Bn, Cn zugeführt. Diese Adressen werden mit den Ausgangssignalen eines Zählers 47 addiert und mit der State machine 45 so verknüpft, daß am Ausgang die neue Zieladresse ansteht. Der Zähler 47 und der Komparat or 48 haben dabei die Aufgabe sicherzustellen, daß jeweils die richtigen Summanden verknüpft werden und daß jeweils am Ende der Zahlenreihen, d. h. bei n = 9 abgebrochen wird. Ist die Addition vollendet, so wird in der Zeitsteuerung 45 ein STOP-Signal generiert und die Schaltung passiv geschaltet. Ebenso kann das STOP-Signal als Eingangssignal für eine Synchronisations-Schal- tung verwendet werden, indem die Synchronisationslo- gik anhand dieses Signals erkennen kann, daß die Ge- samtfunktion "Addieren" gemäß dem nachfolgend be- schriebenen ML1 Programm beendet ist und die MA- CROs somit durch neue ersetzt werden können (zum Beispiel könnte STOP das Signal Sync5 sein).

Der Zeitablauf in der Zeitsteuerung 45 (STATEMA- CHINE) läßt sich dabei wie folgt darstellen, wobei noch anzumerken ist, daß in der Zeitsteuerung 45 eine Verzö- gerungszeit T (in Form von Taktzyklen) zwischen der Adressgenerierung und dem Datenerhalt implementiert ist:

- Im Zyklus 1 wird jeweils der Zähler 47 um 1 erhöht und im Komparator 48 wird geprüft, ob n > 9 erreicht ist; synchron zu diesen Operationen werden die Adressen für A, B, C berechnet;
- im Zyklus (T + 1) werden die Summanden A, B ausgelesen und addiert;
- im Zyklus (T + 2) wird die Summe C abgespei- chert.

Mit anderen Worten heißt dies, daß die Operations- schleife und die eigentliche Addition gerade (T + 2) Taktzyklen erfordert. Im allgemeinen sind für T 2...3 Takte erforderlich, so daß verglichen mit den herkömm- lichen Prozessoren (CPU), die im allgemeinen 50 bis mehrere 100 Taktzyklen benötigen, eine ganz wesent- liche Rechenzeit-Reduzierung möglich wird.

Die anhand von Fig. 11 aufgezeigte Konfiguration soll im folgenden über eine hypothetische MACRO- Sprache ML1 nochmals erläutert werden: Es existieren die Zahlenreihen An und Bn

$$50 \quad \forall n: 0 \leq n \leq 9$$

Es sollen die Summen Ci = Ai + Bi mit $i \in N$ gebildet werden.

53 const n = 9;
array A[n] in RAM[1] at 1000h;
array B[n] in RAM[2] at 0dfa0h;
array C[n] in RAM[1] at 100ah;

for i = 0 to n with (A[i], B[i], C[i])

56 Δ1;
C = Δ1 = A + B;

next;

RAM1 ist der 1. Speicherblock

RAM2 ist der 2. Speicherblock

at folgt die Basisadresse der Arrays

for ist der Schleifenbeginn

next ist das Schleifenende

with () folgen die Variablen, deren Adressen durch die

DE 44 16 881 A1

11

Zählvariable i bestimmt werden

ΔT folgt die Verzögerungszeit für eine Statemachine in Taktzyklen.

Das Timing der Statemachine sieht demnach folgendermaßen aus:

Zyklus Aktivität

- | | |
|-------|--|
| 1 | Zähler erhöhen, Vergleich auf > 9 (ja \Rightarrow Abbruch) und Adressen für A, B, C, berechnen |
| T + 1 | A, B, C, holen und addieren |
| T + 2 | Nach C speichern |

Das heißt — wie bereits erwähnt — die Schleife und die Addition benötigen gerade einmal T + 2 Taktzyklen.

Fig. 11b zeigt den groben Aufbau der einzelnen Funktionen (MACROs) in einem DFP. Die MACROs sind in ihrer etwaigen Lage und Größe eingezeichnet und mit den anhand von Fig. 11a erläuterten entsprechenden Nummern versehen.

Fig. 11c zeigt den groben Aufbau der einzelnen Funktionen auf die RAM-Blöcke 1 und 2: Die Summanden werden nacheinander in aufsteigender Reihenfolge aus den RAM-Blöcken 1 und 2 ab Adresse 1000h beziehungsweise 0dfa0h gelesen und in RAM-Block 1 ab Adresse 100ah gespeichert. Zudem sind die Zähler 47 und 49 gegeben, beide zählen während des Ablaufs der Schaltung von 0 bis 9.

Nach Beendigung des beschriebenen Programms soll ein neues Programm geladen werden, das die Ergebnisse weiterverwertet. Die Umladung soll zur Laufzeit erfolgen. Das Programm ist im Folgenden gegeben:

Es existieren die Zahlenreihen An und Bn, wobei An durch das Ergebnis Cn des vorher ausgeführten Programms gegeben ist:

$$\forall n: 0 \leq n \leq 9$$

Es sollen die Produkte Ci = Ai • Bi mit $i \in N$ gebildet werden.

const n = 9;

array A[n] in RAM[1] at 100ah
array B[n] in RAM[2] at 0dfa0h
array C[n] in RAM[1] at 1015h
for i = 0 to n with (A[i], B[i], C[i])

$\Delta 1$;

C = $\Delta 1 = A \cdot B$;

next;

Die Beschreibung der einzelnen Befehle ist bereits bekannt,

• symbolisiert die Multiplikation.

Die MACRO-Struktur ist in Fig. 18a beschrieben. Fig. 18b gibt in bekannter Weise die Lage und Größe der einzelnen MACROs auf dem Chip an, besonders zu beachten ist die Größe des Multiplizierers 2 in Vergleich zu Addierer 1 aus Fig. 11b. In Fig. 18c ist erneut die Auswirkung der Funktion auf den Speicher aufgezeigt. Zähler 47 zählt erneut von 0 bis 9, d. h. er wird beim Nachladen der MACROs zurückgesetzt.

Besonders zu beachten ist der Zähler 49. Angenommen, das Umladen der MACROs beträgt 10 Taktzyklen. Dann läuft der Zähler 49 von 9 auf 19, da der Baustein dynamisch umgeladen wird, d. h. nur die umzuladenden Teile werden gestoppt, der Rest arbeitet weiter. Das führt nun dazu, daß der Zähler während des Programm-

12

ablaufs von 19 auf 29 hochläuft. (Hiermit soll das dynamische unabhängige Umladen demonstriert werden, in jedem bisher bekannten Baustein würde der Zähler erneut von 0 auf 9 laufen, da er zurückgesetzt wird).

Bei näherer Betrachtung des Problems stellt sich die Frage, warum nicht beide Operationen, die Addition und die Multiplikation in einem Zyklus durchgeführt werden, also die Operation:

Es existieren die Zahlenreihen An und Bn, wobei An durch das Ergebnis von Cn des vorher ausgeführten Programms gegeben ist:

$$\forall n: 0 \leq n \leq 9$$

15 Es sollen die Produkte Ci = (Ai + Bi) • Bi mit $i \in N$ gebildet werden.

path D;

const n = 9;

array A[n] in RAM[1] at 1000h
array B[n] in RAM[2] at 0dfa0h
array C[n] in RAM[1] at 100ah
for i = 0 to n with (A[i], B[i], C[i])

$\Delta 1$;

D = $\Delta 1 = A + B$;

C = $\Delta 1 = D \cdot B$;

next;

path D definiert einen internen nicht aus den DFP herausgeführten Doppelpfad. Die Operation benötigt wegen einem zusätzlichen $\Delta 1$ einen Taktzyklus mehr als vorher, ist insgesamt jedoch schneller als die beiden obigen Programme in Folge ausgeführt, da zum einen die Schleife nur einmal durchlaufen wird, zum zweiten nicht umgeladen wird.

Prinzipiell könnte das Programm auch so formuliert werden:

const n = 9;

array A[n] in RAM[1] at 1000h
array B[n] in RAM[2] at 0dfa0h
array C[n] in RAM[1] at 100ah
for i = 0 to n with (A[i], B[i], C[i])

$\Delta 1$;

C = $\Delta 1 = (A + B) \cdot B$;

next;

Sind die Gatterlaufzeiten des Addierers und des Multiplizierers zusammen kleiner als ein Taktzyklus, kann die Operation $(A + B) \cdot B$ auch in einem Taktzyklus durchgeführt werden, was zu einer weiteren erheblichen Geschwindigkeitssteigerung führt:

const n = 9;

array A[n] in RAM[1] at 1000h
array B[n] in RAM[2] at 0dfa0h
array C[n] in RAM[1] at 100ah
for i = 0 to n with (A[i], B[i], C[i])

$\Delta 1$;

C = $\Delta 1 = (A + B) \cdot B$;

next;

Anhand von Fig. 12 soll ein einfaches Beispiel eines Zellaufbaus erläutert werden. Die Zelle 10 umfaßt zum Beispiel ein UND-Glied 51, ein ODER-Glied 52, ein XOR-Glied 53, einen Inverter 54 sowie eine Registerzelle 55. Die Zelle 10 weist darüberhinaus eingangsseitig zwei Multiplexer 56, 57 mit (den sechzehn Eingängen der Zelle entsprechend Fig. 6) zum Beispiel je sechzehn Eingangsanschlüssen IN1, IN2 auf. Über diesen

60 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 580 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 630 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 650 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 670 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 690 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 710 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 730 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 750 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 770 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 790 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 810 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 830 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 850 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 870 881 882 883 884 885 886 887 888 889 880 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 900 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 920 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 940 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 960 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 980 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1000 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1020 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1040 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064 1065 1066 1067 1068 1069 1060 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083 1084 1085 1086 1087 1088 1089 1080 1091 1092 1093 1094 1095 1096 1097 1098 1099 1090 1101 1102 1103 1104 1105 1106 1107 1108 1109 1100 1111 1112 1113 1114 1115 1116 1117 1118 1119 1110 1121 1122 1123 1124 1125 1126 1127 1128 1129 1120 1131 1132 1133 1134 1135 1136 1137 1138 1139 1130 1141 1142 1143 1144 1145 1146 1147 1148 1149 1140 1151 1152 1153 1154 1155 1156 1157 1158 1159 1150 1161 1162 1163 1164 1165 1166 1167 1168 1169 1160 1171 1172 1173 1174 1175 1176 1177 1178 1179 1170 1181 1182 1183 1184 1185 1186 1187 1188 1189 1180 1191 1192 1193 1194 1195 1196 1197 1198 1199 1190 1201 1202 1203 1204 1205 1206 1207 1208 1209 1200 1211 1212 1213 1214 1215 1216 1217 1218 1219 1210 1221 1222 1223 1224 1225 1226 1227 1228 1229 1220 1231 1232 1233 1234 1235 1236 1237 1238 1239 1230 1241 1242 1243 1244 1245 1246 1247 1248 1249 1240 1251 1252 1253 1254 1255 1256 1257 1258 1259 1250 1261 1262 1263 1264 1265 1266 1267 1268 1269 1260 1271 1272 1273 1274 1275 1276 1277 1278 1279 1270 1281 1282 1283 1284 1285 1286 1287 1288 1289 1280 1291 1292 1293 1294 1295 1296 1297 1298 1299 1290 1301 1302 1303 1304 1305 1306 1307 1308 1309 1300 1311 1312 1313 1314 1315 1316 1317 1318 1319 1310 1321 1322 1323 1324 1325 1326 1327 1328 1329 1320 1331 1332 1333 1334 1335 1336 1337 1338 1339 1330 1341 1342 1343 1344 1345 1346 1347 1348 1349 1340 1351 1352 1353 1354 1355 1356 1357 1358 1359 1350 1361 1362 1363 1364 1365 1366 1367 1368 1369 1360 1371 1372 1373 1374 1375 1376 1377 1378 1379 1370 1381 1382 1383 1384 1385 1386 1387 1388 1389 1380 1391 1392 1393 1394 1395 1396 1397 1398 1399 1390 1401 1402 1403 1404 1405 1406 1407 1408 1409 1400 1411 1412 1413 1414 1415 1416 1417 1418 1419 1410 1421 1422 1423 1424 1425 1426 1427 1428 1429 1420 1431 1432 1433 1434 1435 1436 1437 1438 1439 1430 1441 1442 1443 1444 1445 1446 1447 1448 1449 1440 1451 1452 1453 1454 1455 1456 1457 1458 1459 1450 1461 1462 1463 1464 1465 1466 1467 1468 1469 1460 1471 1472 1473 1474 1475 1476 1477 1478 1479 1470 1481 1482 1483 1484 1485 1486 1487 1488 1489 1480 1491 1492 1493 1494 1495 1496 1497 1498 1499 1490 1501 1502 1503 1504 1505 1506 1507 1508 1509 1500 1511 1512 1513 1514 1515 1516 1517 1518 1519 1510 1521 1522 1523 1524 1525 1526 1527 1528 1529 1520 1531 1532 1533 1534 1535 1536 1537 1538 1539 1530 1541 1542 1543 1544 1545 1546 1547 1548 1549 1540 1551 1552 1553 1554 1555 1556 1557 1558 1559 1550 1561 1562 1563 1564 1565 1566 1567 1568 1569 1560 1571 1572 1573 1574 1575 1576 1577 1578 1579 1570 1581 1582 1583 1584 1585 1586 1587 1588 1589 1580 1591 1592 1593 1594 1595 1596 1597 1598 1599 1590 1601 1602 1603 1604 1605 1606 1607 1608 1609 1600 1611 1612 1613 1614 1615 1616 1617 1618 1619 1610 1621 1622 1623 1624 1625 1626 1627 1628 1629 1620 1631 1632 1633 1634 1635 1636 1637 1638 1639 1630 1641 1642 1643 1644 1645 1646 1647 1648 1649 1640 1651 1652 1653 1654 1655 1656 1657 1658 1659 1650 1661 1662 1663 1664 1665 1666 1667 1668 1669 1660 1671 1672 1673 1674 1675 1676 1677 1678 1679 1670 1681 1682 1683 1684 1685 1686 1687 1688 1689 1680 1691 1692 1693 1694 1695 1696 1697 1698 1699 1690 1701 1702 1703 1704 1705 1706 1707 1708 1709 1700 1711 1712 1713 1714 1715 1716 1717 1718 1719 1710 1721 1722 1723 1724 1725 1726 1727 1728 1729 1720 1731 1732 1733 1734 1735 1736 1737 1738 1739 1730 1741 1742 1743 1744 1745 1746 1747 1748 1749 1740 1751 1752 1753 1754 1755 1756 1757 1758 1759 1750 1761 1762 1763 1764 1765 1766 1767 1768 1769 1760 1771 1772 1773 1774 1775 1776 1777 1778 1779 1770 1781 1782 1783 1784 1785 1786 1787 1788 1789 1780 1791 1792 1793 1794 1795 1796 1797 1798 1799 1790 1801 1802 1803 1804 1805 1806 1807 1808 1809 1800 1811 1812 1813 1814 1815 1816 1817 1818 1819 1810 1821 1

DE 44 16 881 A1

13

xer 58 gekoppelt, der seinerseits mit dem Eingang des Inverters 54, einem Eingang der Registerzelle 55 und einem weiteren (3 : 16)-Multiplexer 59 gekoppelt ist. Der letztgenannte Multiplexer 59 ist zusätzlich mit dem Ausgang des Inverters 54 und einem Ausgang der Registerzelle 55 verbunden und gibt das Ausgangssignal OUT ab.

Der Vollständigkeit halber sei angemerkt, daß die Registerzelle 55 mit einem Reset-Eingang R und einem Takteingang gekoppelt ist.

Dem im vorstehenden erläuterten Zellenaufbau, d. h. der Zelle 10 ist nun eine Ladelogik 30 übergeordnet, die mit den Multiplexern 56, 57, 58 und 59 verbunden ist und diese den gewünschten Funktionen entsprechend ansteuert.

Sollen zum Beispiel die Signale A2 mit B5 verundet werden, so werden die Multiplexer 56, 57 den Leitungen "ZWEI" beziehungsweise "FÜNF" entsprechend aktiv geschaltet; die Summanden gelangen dann zum UND-Glied 51 und werden bei entsprechender Aktivierung der Multiplexer 58, 59 am Ausgang OUT abgegeben. Soll zum Beispiel eine NAND-Verknüpfung durchgeführt werden, so schaltet der Multiplexer 58 zum Inverter 54 und am Ausgang OUT steht dann das negierte UND-Ergebnis an.

Patentansprüche

1. Datenverarbeitungseinrichtung, wobei ein (im folgenden Datenflußprozessor — DFP — genannt) integrierter Schaltungskreis (Chip) mit einer Vielzahl insbesondere orthogonal zueinander angeordneter homogen strukturierter Zellen mit je einer Mehrzahl jeweils logisch gleicher und strukturell identisch angeordneter Bausteine vorgesehen ist, dessen Zellen zeilen- und spaltenweise, gegebenenfalls gruppenweise zusammengefaßt, mit Ein-/Ausgangsanschlüssen des integrierten Schaltungskreises verbunden sind, dadurch gekennzeichnet, daß den Zellen eine Ladelogik zugeordnet ist, über die sie je für sich und gegebenenfalls gruppenweise zusammengefaßt so programmierbar (konfigurierbar) sind, daß beliebige logische Funktionen und/oder Vernetzungen untereinander verifizierbar sind, und zwar derart, daß eine Manipulation der DFP-Konfiguration während des Betriebes (oder zur Laufzeit), d. h. die Modifikation funktioneller Teile (MACROS) des DFPs erfolgen kann, ohne daß andere funktionelle Teile angehalten werden müssen oder in ihrer Funktion beeinträchtigt werden.

2. Datenverarbeitungseinrichtung nach Anspruch 1, dadurch gekennzeichnet, daß die Ladelogik mit Speichermitteln gekoppelt ist, über die die Konfiguration der Zellen spezifizierbar ist.

3. Datenverarbeitungseinrichtung nach einem der Ansprüche 1 bis 2, dadurch gekennzeichnet, daß die Ladelogik aus einem Prozessor besteht, der den gesamten Programmablauf auf der Grundlage von in verschiedenen Speichern abgelegten Daten und Programmen im Sinne einer Harvard-Struktur verwaltet.

4. Datenverarbeitungseinrichtung nach Anspruch 3, dadurch gekennzeichnet, daß die Ladelogik ihrerseits aus Zellen mit je einer Mehrzahl jeweils logisch gleicher und strukturell identisch angeordneter Bausteine aufgebaut ist.

5. Datenverarbeitungseinrichtung nach einem der

14

Ansprüche 1 oder 4, dadurch gekennzeichnet, daß die Zellen dynamisch während eines Programmablaufs umkonfigurierbar sind, ohne daß die zu bearbeitenden Daten selbst beeinflußt werden.

6. Datenverarbeitungseinrichtung, wobei ein (im folgenden Datenflußprozessor — DFP — genannt) integrierter Schaltungskreis (Chip) mit einer Vielzahl insbesondere orthogonal zueinander angeordneter homogen strukturierter Zellen mit je einer Mehrzahl jeweils logisch gleicher und strukturell identisch angeordneter Bausteine vorgesehen ist, dessen Zellen zeilen- und spaltenweise, gegebenenfalls gruppenweise zusammengefaßt, mit Ein-/Ausgangsanschlüssen des integrierten Schaltungskreises verbunden sind, dadurch gekennzeichnet, daß eine Mehrzahl von ihnen in Kaskadenform koppelbar sind.

7. Datenverarbeitungseinrichtung nach einem der Ansprüche 1 bis 6, gekennzeichnet durch, die Zuordnung geeigneter Daten-Ein-/Ausgabe-Einheiten und mindestens eines Speichers zum Aufbau eines (komplexen, kompletten) Rechenwerks.

8. Datenverarbeitungseinrichtung nach einem der Ansprüche 1 bis 7, dadurch gekennzeichnet, daß die Funktionen der Ein-/Ausgabeeinheiten partiell im integrierten Schaltungskreis (Chip) implementierbar sind.

Hierzu 11 Seite(n) Zeichnungen

07/20 14:55 1998

07/20/98 14:55 FAX 18008881323

FAXPAT INC

010

- Leerseite -

ZEICHNUNGEN SEITE 1

Nummer: DE 44 16 881 A1
 Int. Cl. 5: G 06 F 15/80
 Offenlegungstag: 17. November 1994

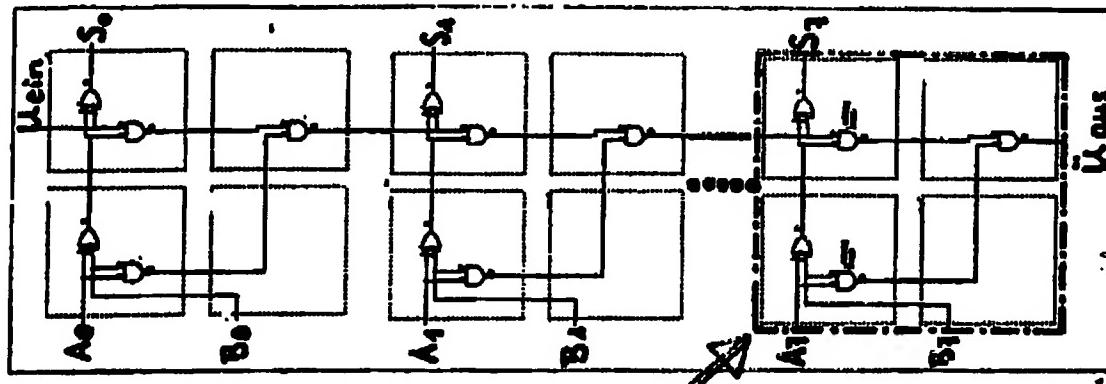


Fig. 5

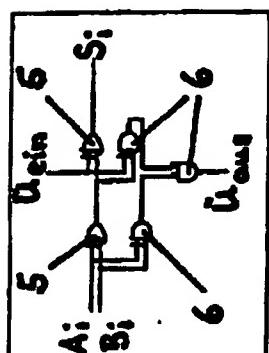


Fig. 3

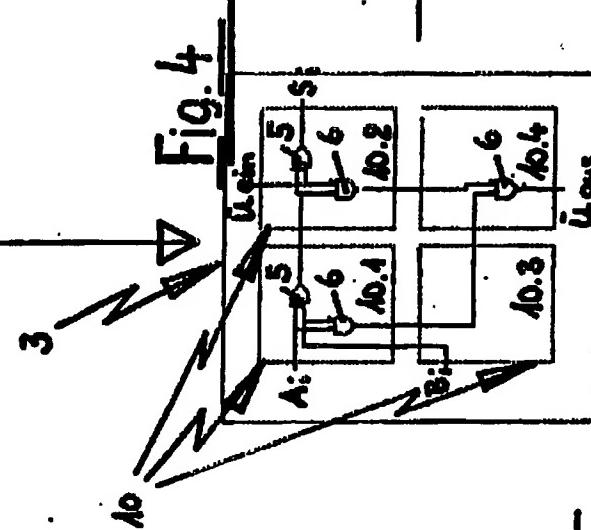


Fig. 4

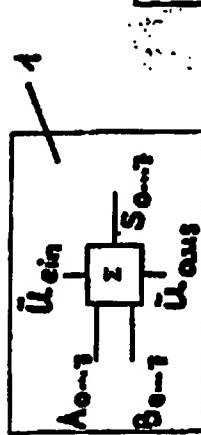


Fig. 1

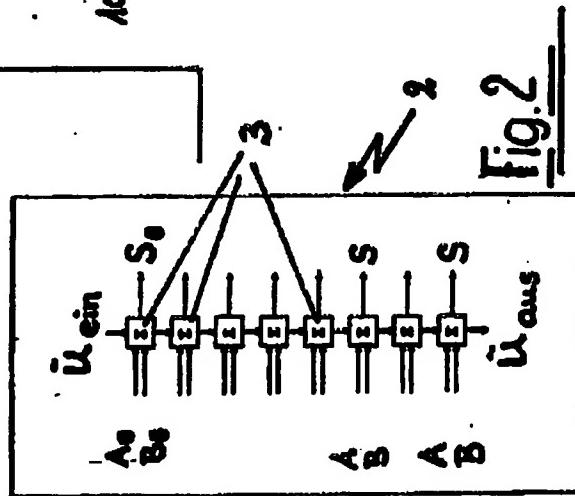
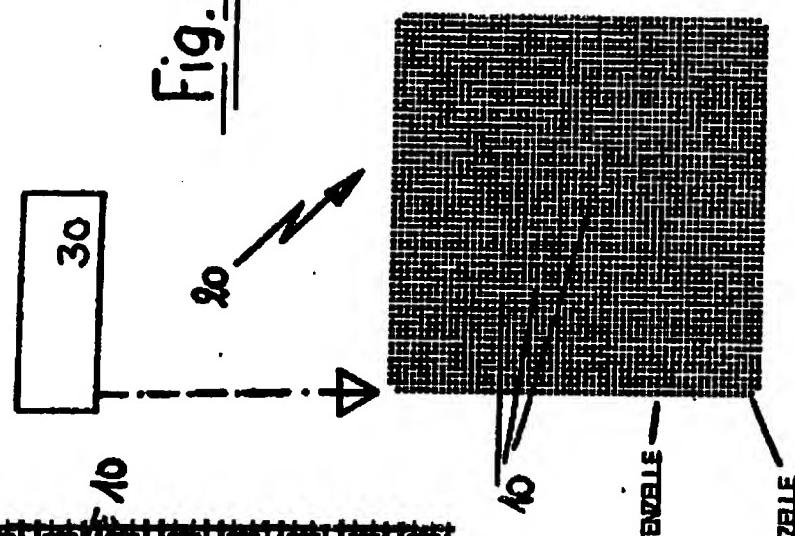
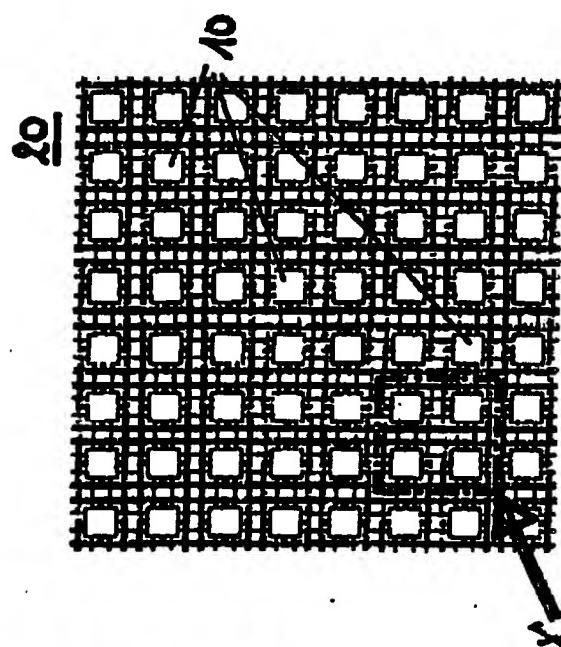
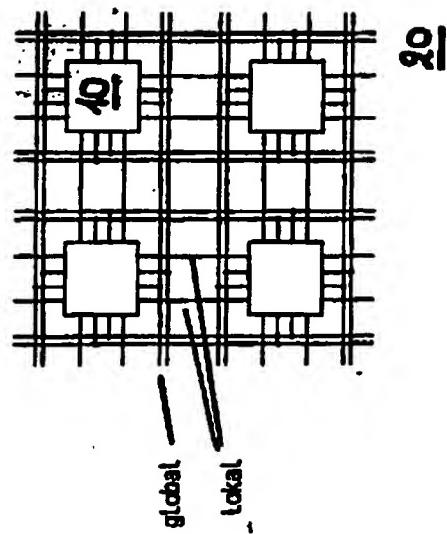


Fig. 2

ZEICHNUNGEN SEITE 2

Nummer:
Int. Cl. 5:
Offenlegungstag:

DE 44 16 881 A1
G 06 F 15/80
17. November 1994

Fig. 8Fig. 7Fig. 6

ZEICHNUNGEN SEITE 3

Nummer:

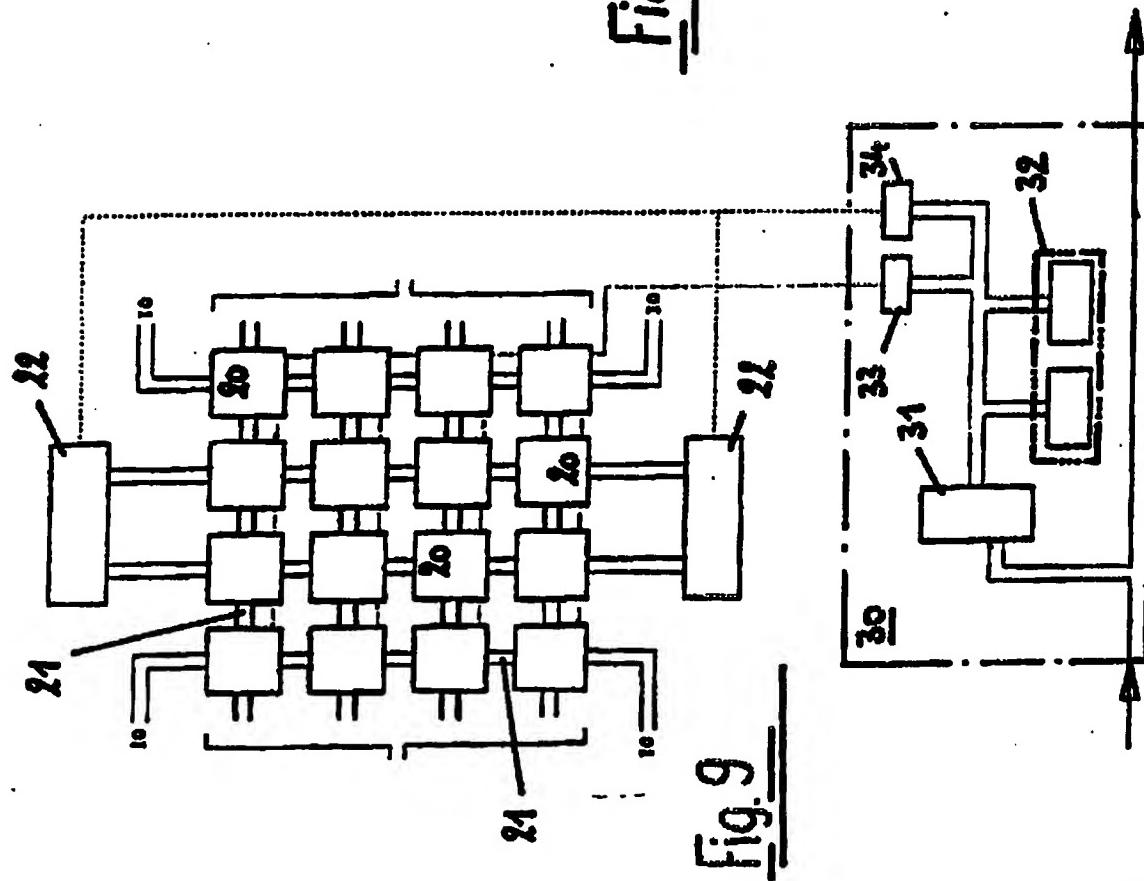
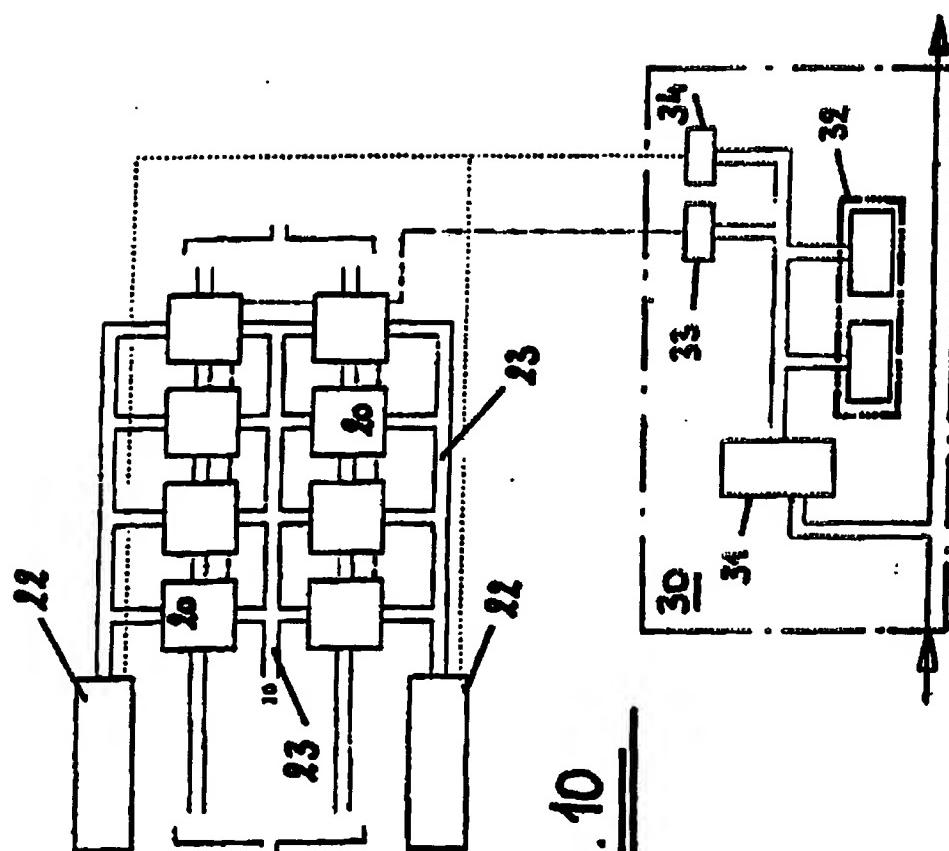
DE 44 16 881 A1

Int. Cl.⁵:

G 08 F 15/80

Offenlegungstag:

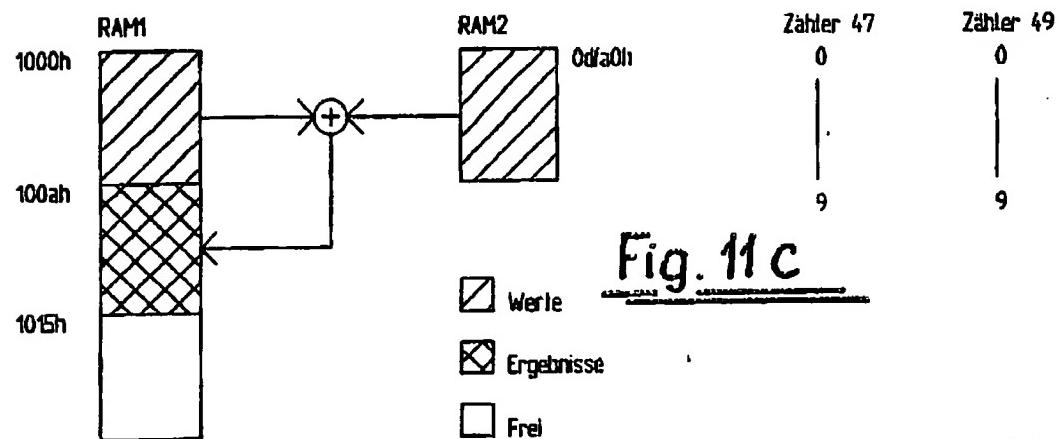
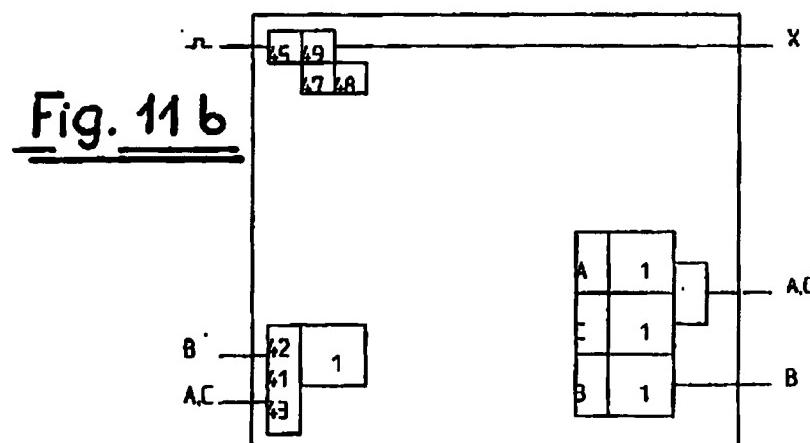
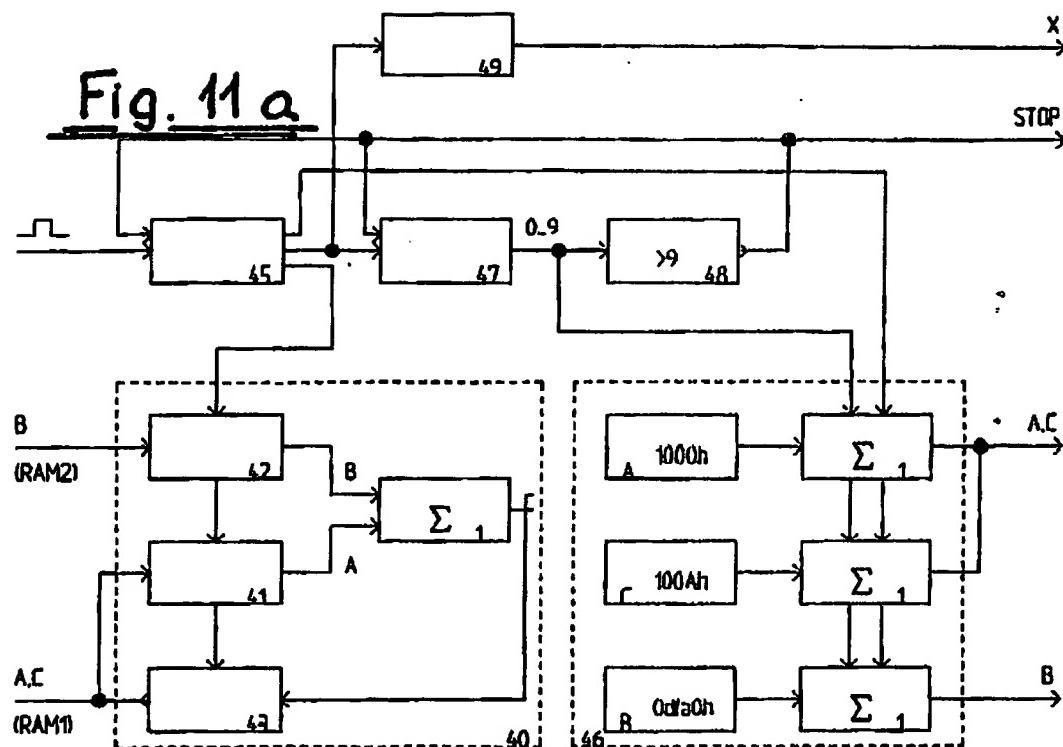
17. November 1994



ZEICHNUNGEN SEITE 4

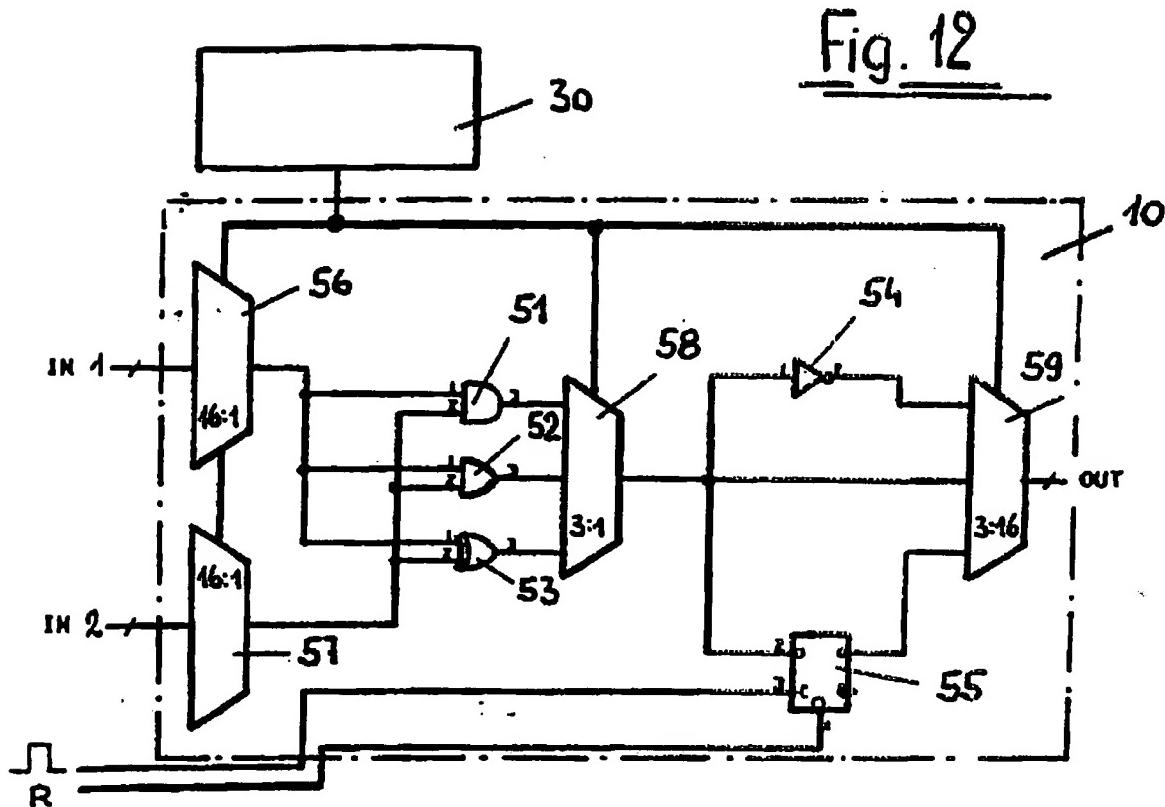
Nummmer:
Int. Cl. 5:
Offenlegungstag:

DE 44 16 881 A1
G 08 F 15/80
17. November 1994



ZEICHNUNGEN SEITE 5

Nummer: DE 44 16 881 A1
 Int. Cl. 8: G 06 F 15/80
 Offenlegungstag: 17. November 1984



DFP-interne
Sync-Signale
aus den einzelnen
MACROS

| TTL148 | |
|--------|--------|
| Sync0 | 1 |
| Sync1 | 2 |
| Sync2 | 3 |
| Sync3 | 4 |
| Sync4 | 5 |
| Sync5 | 6 |
| Sync6 | 7 |
| | 0 |
| | 1 |
| | 2 |
| | LSync0 |
| | LSync1 |
| | LSync2 |

zur Ladelogik

Fig. 13

ZEICHNUNGEN SEITE 6

Nummer:
Int. Cl.⁵:
Offenlegungstag:

DE 44 18 881 A1
G 08 F 15/80
17. November 1994

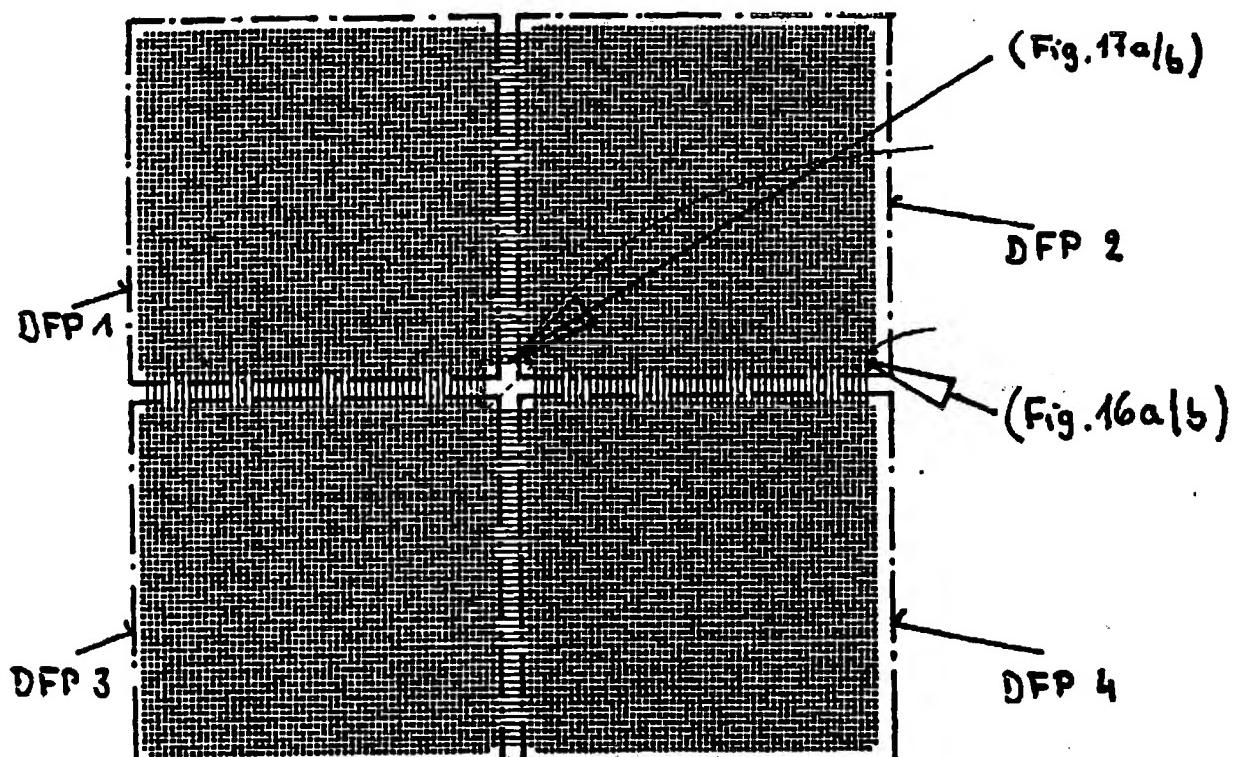


Fig. 14

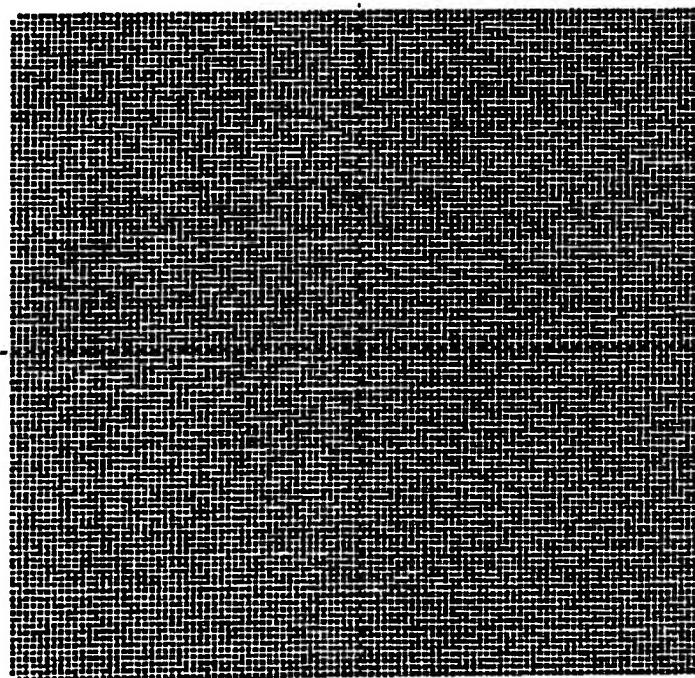
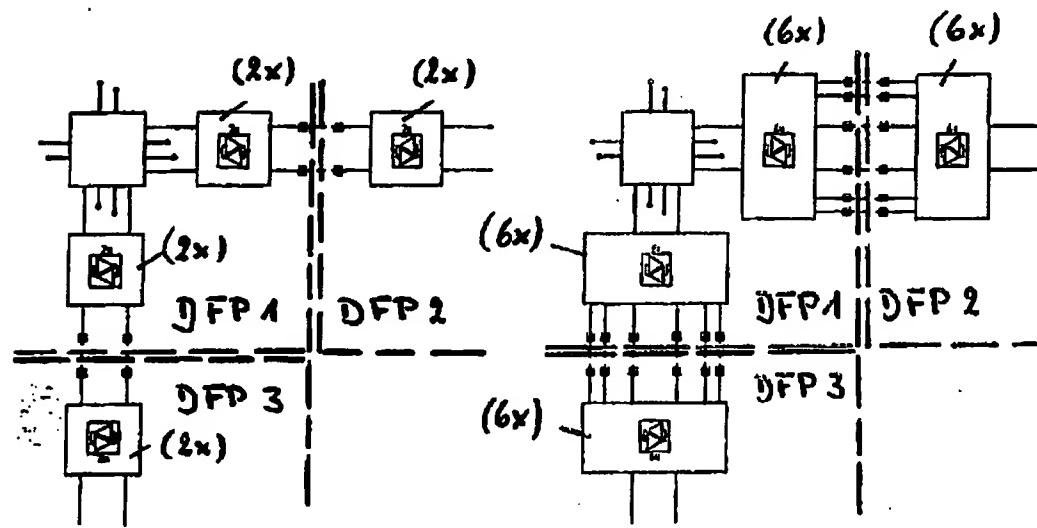
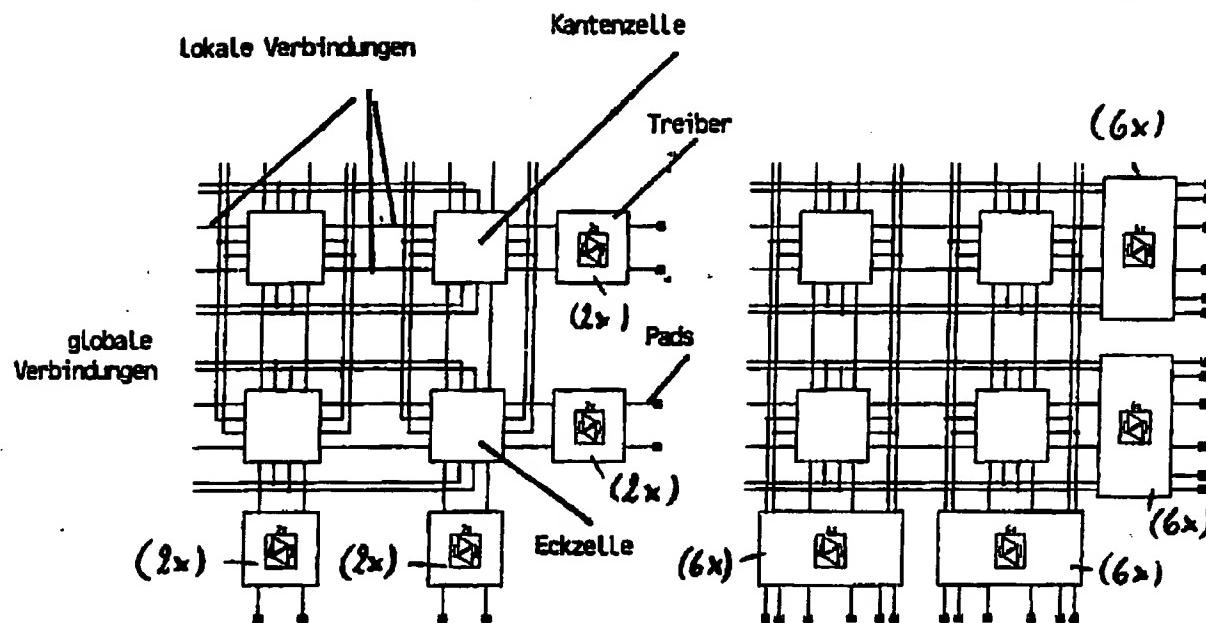


Fig. 15

ZEICHNUNGEN SEITE 7

Nummer:
Int. Cl. 5:
Offenlegungstag:

DE 44 16 881 A1
G 06 F 15/80
17. November 1994



ZEICHNUNGEN SEITE 8

Nummer:
Int. Cl. 5:
Offenlegungstag:

DE 44 16 881 A1
G 06 F 15/80
17. November 1994

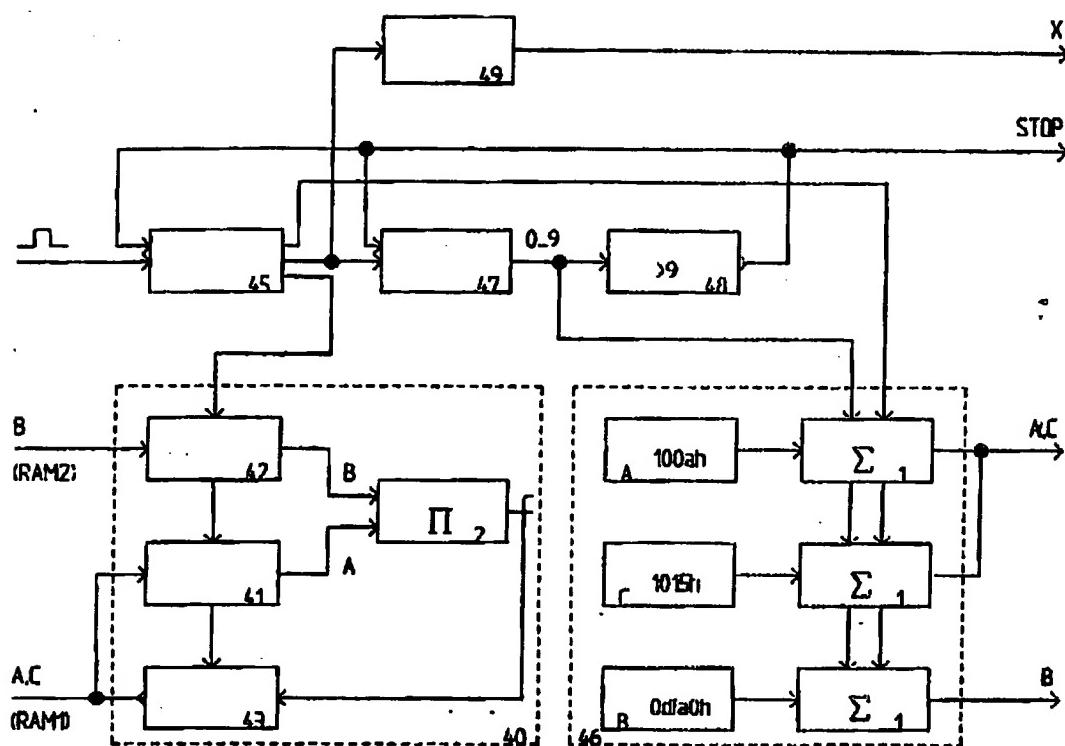


Fig. 18a

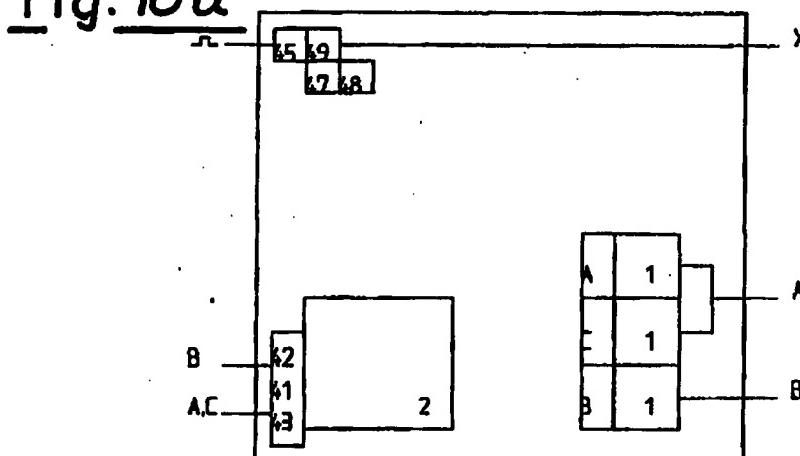


Fig. 18 b

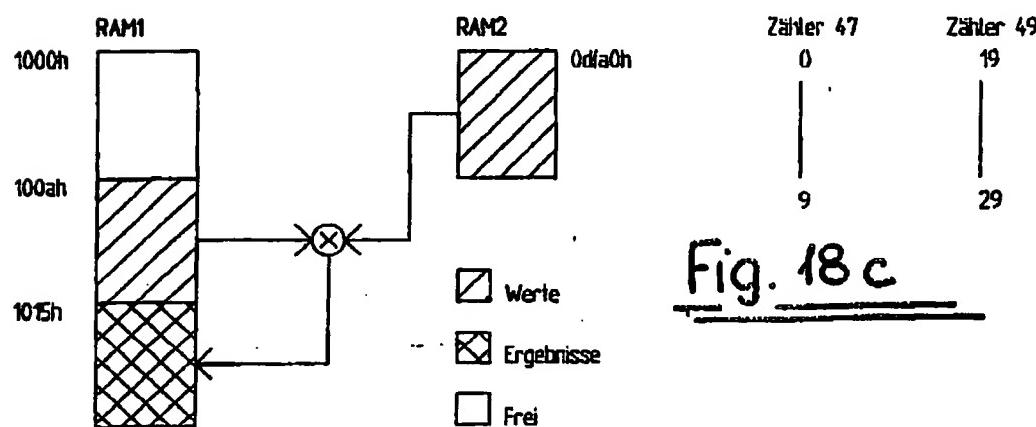
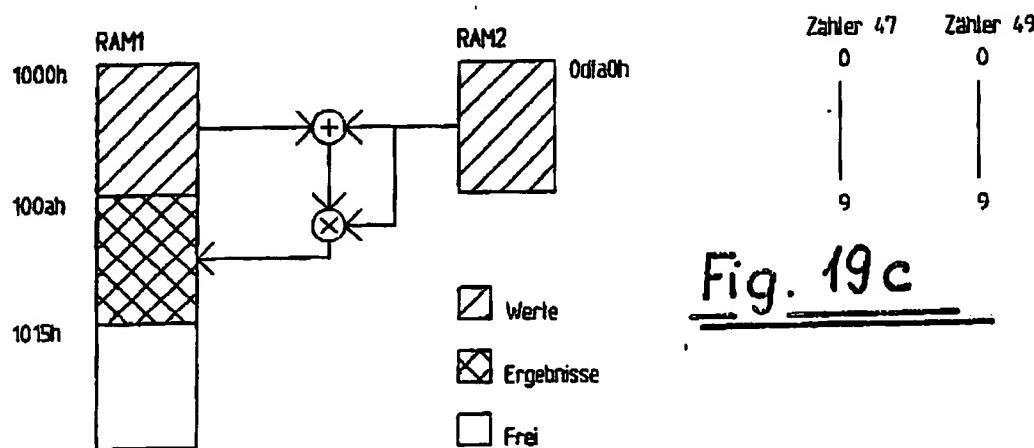
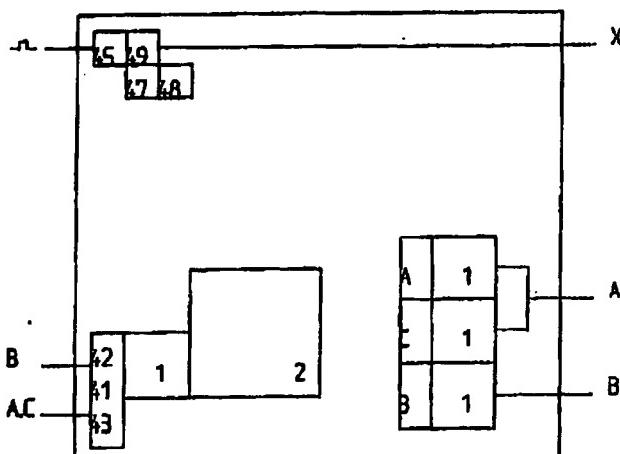
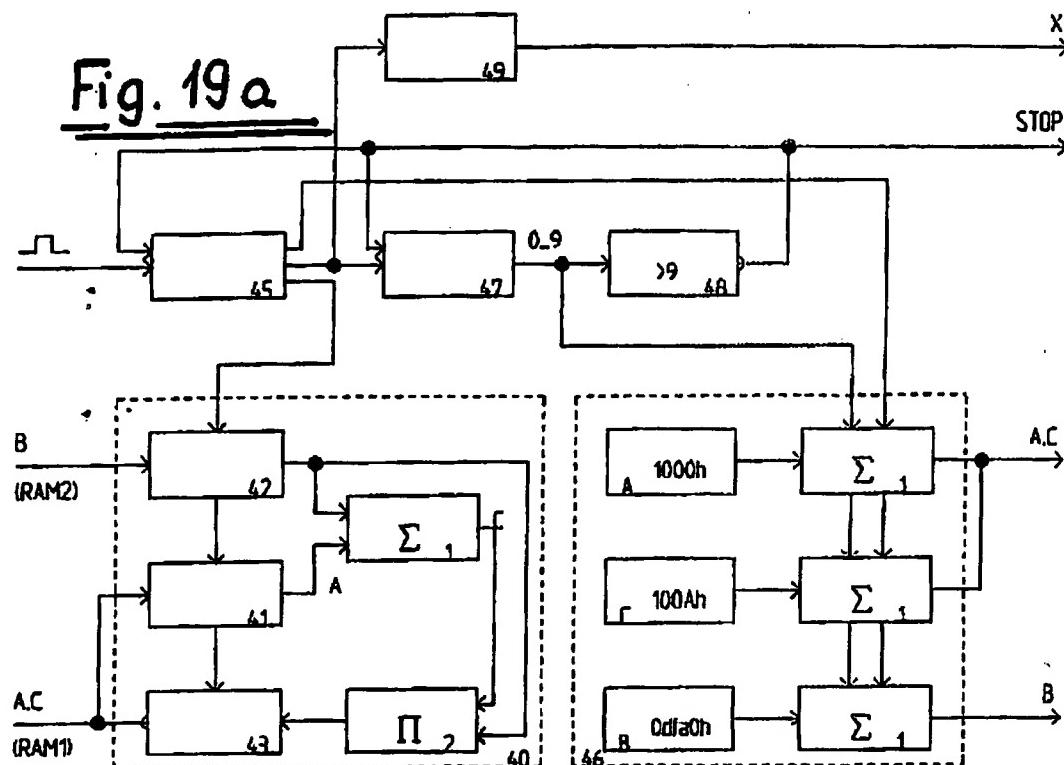


Fig. 18 c

ZEICHNUNGEN SEITE 9

Nummer:
Int. Cl. 5:
Offenlegungstag:

DE 44 16 881 A1
G 08 F 15/80
17. November 1994



ZEICHNUNGEN SEITE 10

Nummer:

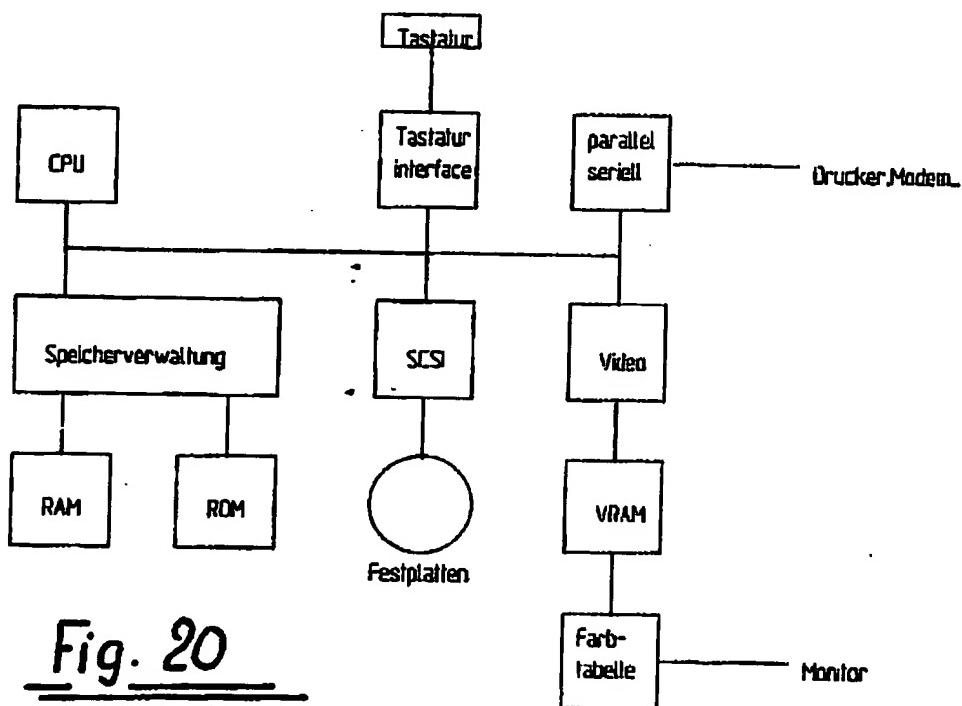
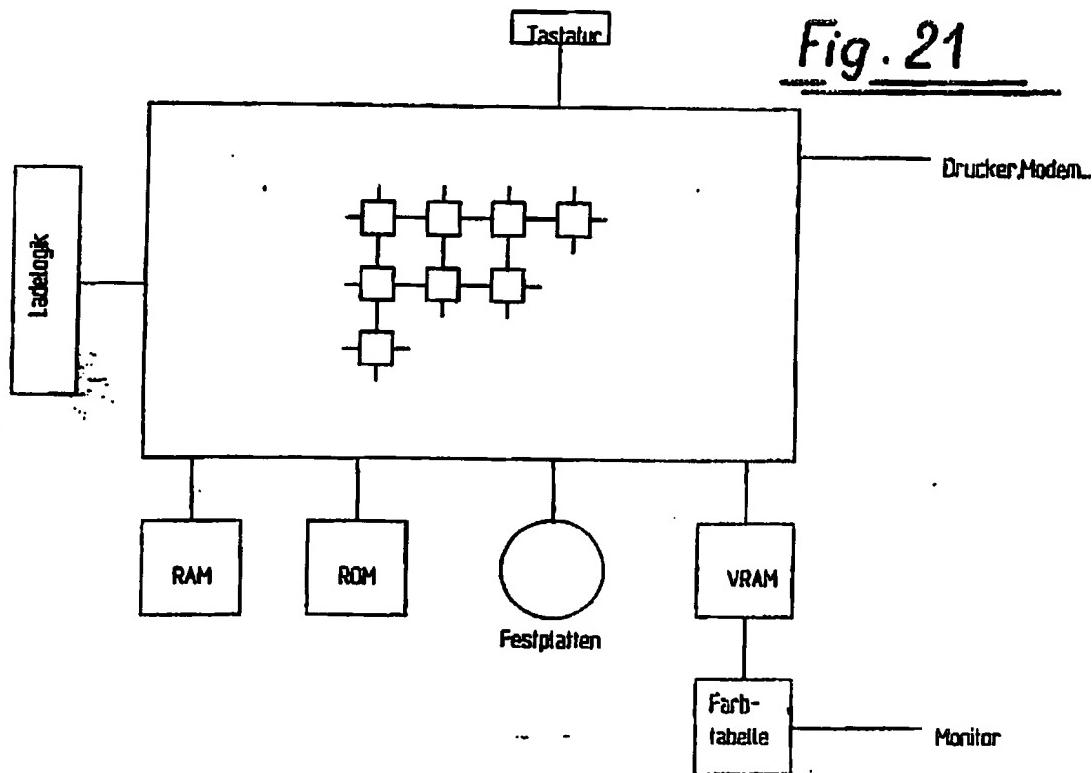
DE 44 18 881 A1

Int. Cl. 5:

G 06 F 15/80

Offenlegungstag:

17. November 1994

Fig. 20Fig. 21

ZEICHNUNGEN SEITE 11

Nummer: DE 44 16 881 A1
Int. Cl. 5: G 08 F 15/80
Offenlegungstag: 17. November 1994

Fig. 22

